INTERMICRO Collection dirigée par Jacky Akoka

Votre ORIC

Oric/Oric Atmos

Ian Mc Lean

PHI

Votre ORIC

Oric/Oric Atmos

Pour Anne

Ian Mc Lean

Votre ORIC

Oric/Oric Atmos

Prentice/Hall PHI International

Inter Editions

87 avenue du Maine
75014 Paris

L'édition originale de cet ouvrage a été publiée en Grande-Bretagne par Prentice/Hall International, Londres, sous le titre *Understanding Oric,* © 1984 by Ian McLean.

Prentice-Hall International, Inc., *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall Canada, Inc., *Toronto*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte., *Singapore*
Whitehall Books Limited, Wellington, *New Zealand*
Prentice-Hall Inc., Englewood Cliffs, *New Jersey*

ISBN 0-13-936980-5

Edition française
© 1984, InterEditions, Paris et Prentice-Hall International. Tous droits réservés. Aucun extrait de ce livre ne peut être reproduit, sous quelque forme ou par quelque procédé que ce soit (machine électronique, mécanique, à photocopier, à enregistrer ou toute autre) sans l'autorisation écrite préalable des Editeurs.

ISBN 2-7296-0112-0

SOMMAIRE

PREFACE X

NOTE DU DIRECTEUR DE COLLECTION XIII

CHAPITRE 1 1

ORIC EN FONCTIONNEMENT

DEBALLAGE DE L'ORDINATEUR/CONNECTION/COMMENCONS/ORIC LE BRUYANT/DE TOUTES LES COULEURS/NETTOYER L'ECRAN/ ENLEVER LE CLIQUETIS DES TOUCHES/L'INTERRUPTEUR RESET/ QUELQUES TRUCS AMUSANTS/QUELQUES BELLES IMAGES.

CHAPITRE 2 24

LES TOUCHES DU SUCCES

LE CLAVIER/TOUCHES SHIFTEES/TOUCHE D'EFFACEMENT/RETURN/ REPETITION AUTOMATIQUE/LE TAMPON DU CLAVIER/LETTRES MINUSCULES/CARACTERES DE CONTROLE/ATTRIBUTS/AVANT-PLAN ET ARRIERE-PLAN/CTRL]/CLIGNOTANT ET PERMANENT/ CARACTERES STANDARDS ET CARACTERES ALTERNATIFS/ CARACTERES DOUBLE HAUTEUR/ATTRIBUTS FONCTIONNANT AVEC 60HZ.

CHAPITRE 3 37

ENTREZ DANS L'ORIC

ENTREE DE PROGRAMMES/NUMEROS DE LIGNE/ENTREE DE PROGRAMMES/MODIFICATIONS/NUMEROTATION DES LIGNES/ SUPPRESSION D'UNE LIGNE/NEW/UNE RECREATION.

CHAPITRE 4 51

CHARGEMENT ET SAUVEGARDE DES PROGRAMMES

ENREGISTREUR DE CASSETTES/CASSETTES/TERMES TECHNIQUES/ CSAVE/CLOAD/PROBLEMES RELATIFS AUX ENREGISTREURS DE CASSETTES/RAPIDE ET LENT/VERIFICATION DE PROGRAMMES/ FUSION DE PROGRAMMES/ RECHERCHE DES PROGRAMMES/ DEMARRAGE AUTOMATIQUE/SAUVEGARDE DES BLOCS DE MEMOIRE/SAUVEGARDE DE LONGS PROGRAMMES/ENCORE QUELQUES RECREATIONS.

CHAPITRE 5 64

EXPLORATION DE L'ORIC

CARACTERISTIQUES GENERALES/REM/DEUX POINTS (:)/STOP/ CONT/GOTO/BOUCLE INFINIE/DEFILEMENT/CLS/FOR...NEXT/ LET/WAIT/INK ET PAPER.

CHAPITRE 6 74

L'AFFICHAGE DE L'ORIC

PRINT/IMPRESSION DE NOMBRES/IMPRESSION DE CARACTERES/ CHAINES DE CARACTERES/LES SEPARATEURS DU PRINT/POINT VIRGULE/VIRGULE/ SPC/PRINT @/ARITHMETIQUE.

CHAPITRE 7 89

QUELQUES CARACTERES BIZARRES

CHR$/CARACTERES DE CONTROLE/ESCAPE/PLOT/LORES 0/ ATTRIBUTS DE L'INSTRUCTION PLOT/ATTRIBUTS AVEC PRINT@/INPUT.

CHAPITRE 8 103

IL FAUT PRENDRE UNE DECISION

DECISIONS/IF...THEN/AND, OR, NOT/ IF...THEN...ELSE/ LIGNES CONTENANT DES INSTRUCTIONS MULTIPLES/ FOR...NEXT/STEP/REPEAT...UNTIL/SOUS-PROGRAMMES/ ON...GOTO/ON...GOSUB/TRON ET TROFF.

CHAPITRE 9 117

QUELQUES MOTS SUR LES MOTS

CHAINES/CONCATENATION/FRE/COMPARAISON DES CHAINES DE CARACTERES/VAL/STR$/LEN/LEFT$/RIGHT$/MID$/KEY$/GET/ASC/CHR$/NOM DE VARIABLES/CLEAR.

CHAPITRE 10 128

DES NOMBRES AMUSANTS

LES NOMBRES/SGN/ABS/INT/RND/MATHEMATIQUES/PUISSANCE/SQR/LOG/EXP ET LN/PI/SIN, COS, TAN, ATN/DEF FN/READ, DATA, RESTORE/ TABLEAUX/DIM/DES NOMBRES UN PEU DROLES/AND, OR.

CHAPITRE 11 142

LE SON DE L'ORDINATEUR

PLAY/SOUND/ENVELOPPE/MUSIC/THEORIE MUSICALE/HARMONIE/HAUTEUR DU SON/DUREE/MESURE/METHODE DE PROGRAMMATION.

CHAPITRE 12 163

ORIC L'ARTISTE

GRAPHIQUES EN BASSE RESOLUTION/SELECTION DES CARACTERES ALTERNATIFS/LORES 1/GRAPHIQUES DEFINIS PAR L'UTILISATEUR/POKE ET PEEK/ANIMATION/SCRN.

CHAPITRE 13 177

CARACTERISTIQUES ARTISTIQUES SUPPLEMENTAIRES D'ORIC

HIRES/ECRAN HAUTE RESOLUTION/CURSET/CURMOV/DRAW/COULEUR GLOBALE DANS LE MODE HAUTE RESOLUTION/CIRCLE /PATTERN/POINT/FILL/ATTRIBUTS EN MODE HAUTE RESOLUTION/ATTRIBUTS DE L'INSTRUCTION POKE/CHAR/CARACTERES DEFINIS PAR L'UTILISATEUR/DESSINS DE COURBES.

CHAPITRE 14 201

QUELQUES EXPLICATIONS SUPPLEMENTAIRES

MANIPULATION DES DONNEES/GESTION DE FICHIERS - STORE ET RECALL/GRAB/RELEASE/SAUVEGARDE D'AFFICHAGE A L'ECRAN/SAUVEGARDE D'ENSEMBLE DE CARACTERES/TRUE (VRAI) ET FALSE (FAUX)/POP/POS/ PULL/ERREURS DANS LES TABLEAUX ET DANS LES CHAINES DE CARACTERES/ PERIPHERIQUES D'ENTREES ET DE SORTIES/TOUT SUR LES ATTRIBUTS.

CHAPITRE 15 217

COMMENCEZ A IMPRIMER

L'IMPRIMANTE D'ORIC/POSITION DE LA PLUME/MODE D'IMPRESSION/MODE TEXTE/CARACTERES PAR LIGNE/LLIST/ LPRINT/CODES DE CONTROLE/MODE GRAPHIQUE/COMMANDES IMPLICITES/AUTRES IMPRIMANTES/CONNECTION DES BROCHES/ IMPRIMANTE COULEUR TRS-80/POS(1).

CHAPITRE 16 230

INSTRUCTIONS MACHINE

INSTRUCTIONS MACHINE/LE MICROPROCESSEUR 6502/LANGAGE ASSEMBLEUR/CALL/RTS/COMMENT METTRE DES CODES MACHINES EN MEMOIRE/HIMEM/DEEK ET DOKE/INSTRUCTION DEFINIE PAR L'UTILISATEUR (!)/FONCTION DEFINIE PAR L'UTILISATEUR/ CHARGEMENT ET SAUVEGARDE D'INSTRUCTIONS EN LANGAGE MACHINE.

CHAPITRE 17 241

EN GUISE DE CONCLUSION

ANNEXE A 243

CODES ASCII

ANNEXE B 246

BASIC ORIC

ANNEXE C 262

OCCUPATION MEMOIRE

ANNEXE D 264

DESCRIPTION DES BROCHES DES PRISES

ANNEXE E 265

MESSAGES D'ERREURS

ANNEXE F 267

ENSEMBLE DES INSTRUCTIONS EN ASSEMBLEUR 6502

INDEX 285

INDEX

PREFACE

"Cher Monsieur,

Nous regrettons les quelques erreurs qui ont été faites à propos de votre compte. Celles-ci n'étaient pas imputables à certaines personnes de notre organisation car elles étaient causées par notre ordinateur."

Ceci est l'extrait authentique d'une lettre envoyée à un certain nombre de personnes par le département comptable d'une société internationale bien connue. Ce qui est encore plus surprenant est que la plupart des gens à qui cette lettre fut envoyée acceptèrent l'excuse comme valable.

L'ordinateur a une très mauvaise image auprès du public, qui n'est pas améliorée par les récits de la presse populaire. L'ordinateur (à ce qu'il paraît) est une machine féroce, totalement démoniaque et bien plus intelligente que les pauvres personnes sur lesquelles elle fond. Sa principale occupation est d'envoyer des factures d'électricité pour des montants de millions de francs à des personnes âgées. Pendant ses temps libres elle cause des perturbations dans le traffic et est responsable d'à peu près tous les maux qui affligent notre civilisation moderne ; depuis le brouillard jusqu'à la surpopulation (ici nous nous interrogeons).

Les seuls êtres autorisés à pénétrer dans l'antre climatisé du monstre sont une élite spéciale de génies qui doivent avoir des doctorats dans au moins quatre domaines différents. Tout autre mortel inférieur ayant l'audace d'approcher l'ordinateur ou encore de toucher le clavier sera immédiatement volatilisé. Toute entrée non autorisée au clavier fera que l'ordinateur pourra faire exploser le Monde.

N'en croyez pas un mot.

L'ordinateur est juste une machine comme toute autre. Il fera seulement ce que vous lui dites de faire. Il n'est pas très intelligent et fera exactement ce qu'on lui dit de faire et non pas ce que vous pensez que vous lui avez dit de faire. Il ne fera pas ce qu'on ne lui dit pas de faire. Même si cela vous paraît logique que c'est cela qu'il fallait faire. Vous verrez ce que je veux dire une fois que vous commencerez à programmer l'ordinateur (lui dire ce qu'il faut faire).

VOUS AVEZ LE CONTROLE DE LA SITUATION

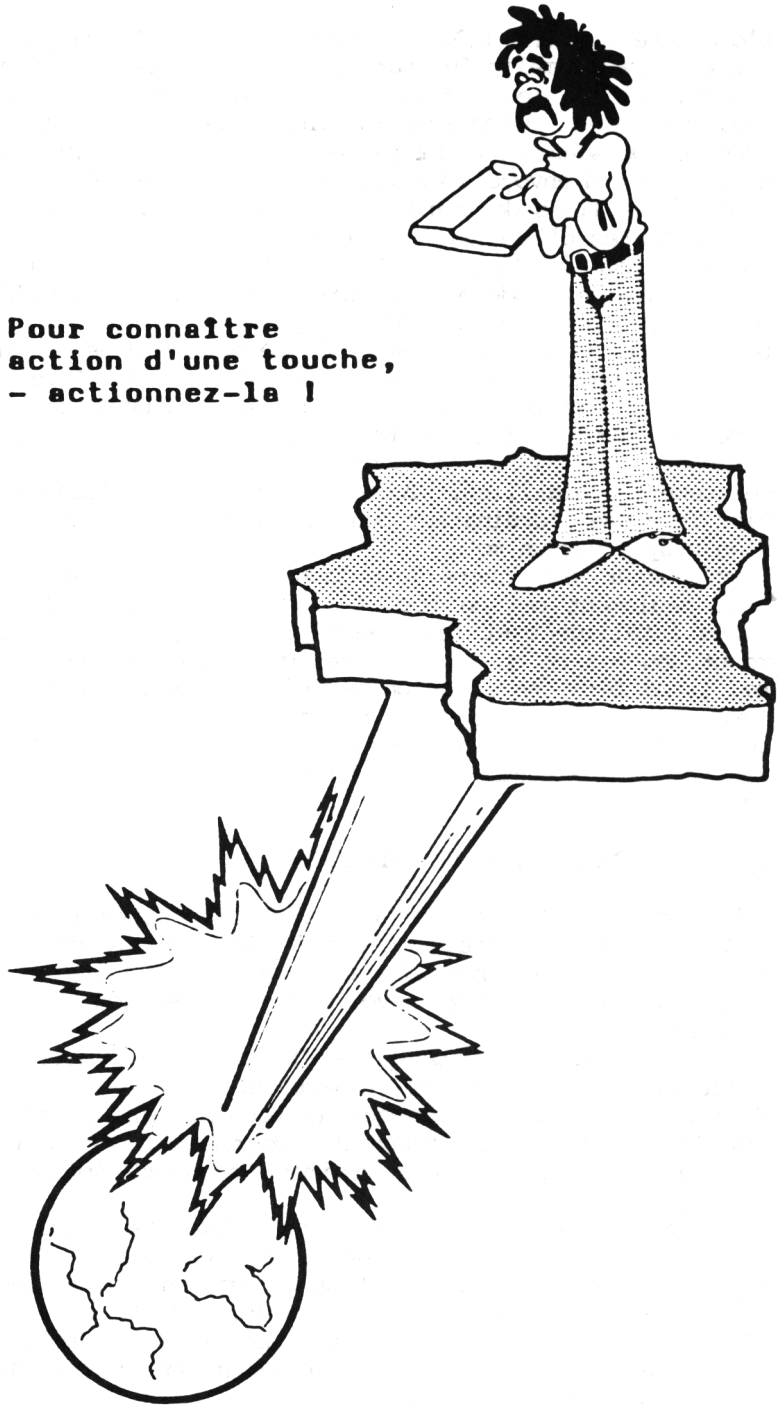
C'est maintenant l'ère de l'ordinateur personnel. ORIC est votre propre propriété. C'est à vous de lui dire quoi faire, et sans vos instructions il ne peut rien faire.

Aussi si vous voulez savoir ce qu'une touche fait, pressez-là.

Si vous voulez savoir ce qui arrivera si vous mettez une instruction dans un programme, essaye-le.

Vous ne pouvez pas endommager la machine par des entrées au clavier ou des instructions de programme ; s'asseoir et jouer avec la machine c'est vraiment amusant. Amusez-vous bien.

S'il vous plaît considérez les programmes que vous trouverez dans ce livre comme des exemples. Ils ne représentent pas la seule solution et peuvent très bien ne pas être la meilleure solution. Changez-les de la manière qu'il vous plaira. En ce faisant vous aurez une meilleure compréhension de la manière dont le programme fonctionne et comment la machine travaille.



dessin de Tony Hinwood, 'Graphics Ad Lib No.3', Business Books

NOTE DU DIRECTEUR DE COLLECTION

Toutes mes félicitations ! En achetant ce livre vous montrez votre intérêt de tirer profit de l'ORIC ou de l'ORIC-ATMOS.

Cet extraordinaire microordinateur comporte en effet de multiples facettes avec des possibilités insoupçonnées tant au niveau de la programmation (BASIC, Assembleur, etc.) que logiciels d'application (gestionnaire de fichiers, traitement de texte, etc.). En outre, il vous offre d'énormes possibilités en créations graphiques et sonores.

Un seul conseil à vous donner : suivez pas à pas les recommendations de l'auteur et, surtout, réalisez systématiquement tous les exemples réalisés dans cet ouvrage. Vous serez étonné des progrès que vous réaliserez.

Bon travail et surtout bonne détente !

J. Akoka
Juin 1984

Chapitre 1

ORIC EN FONCTIONNEMENT

"Maintenant qu'il était maître du
Monde, il n'était pas tout à fait sûr
de ce qu'il devait faire ensuite.

Mais il penserait à quelque chose."

Arthur C. Clarke
2001

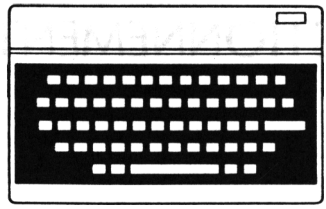
Félicitations !

Vous possédez maintenant un ORIC. Probablement un des meilleurs achats actuels en matière d'ordinateur familial. Vous avez envoyé les enfants au lit, dit à tous vos amis que vous partiez pour la Mongolie Extérieure, et mis le chat dehors pour jouer avec le chien de berger du voisin.

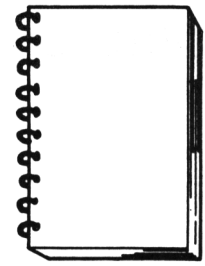
Vous êtes maintenant seul avec votre machine.

DEBALLAGE DE L'ORDINATEUR

Ouvrez soigneusement la boîte et enlevez l'emballage protecteur. Vous devez avoir les éléments suivants :

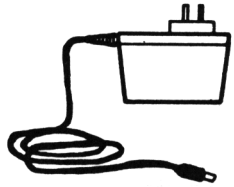


1 Ordinateur



1 Manuel Utilisateur

1 Cable de Branchement TV



1 Alimentation Electrique

1 Cassette de Bienvenue

1 Cable de Branchement Magnétophone

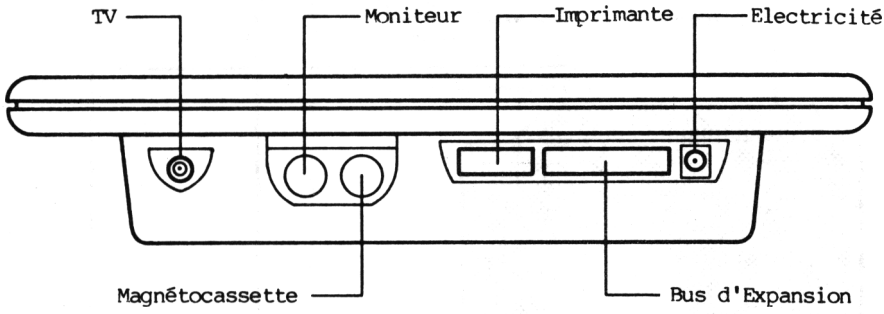
De manière à faire fonctionner votre ordinateur vous avez besoin d'un poste de télévision. L'ordinateur travaillera aussi bien avec une télévision noir et blanc qu'avec une télévision couleur. Tous les programmes dans ce livre fonctionneront quelque soit le téléviseur que vous possédiez. Les programmes utilisant la couleur produiront différentes tonalités de gris sur un récepteur noir et blanc.

Toutefois, je souhaite que vous ayez un téléviseur couleur. Des affichages spectaculaires peuvent être produits par l'ORIC sur celui-ci.

Pour le reste du livre, je considérerai que le lecteur a une télévision couleur.

CONNECTION

Si vous regardez à l'arrière de l'ordinateur vous verrez un certain nombre de fiches comme celles-ci :

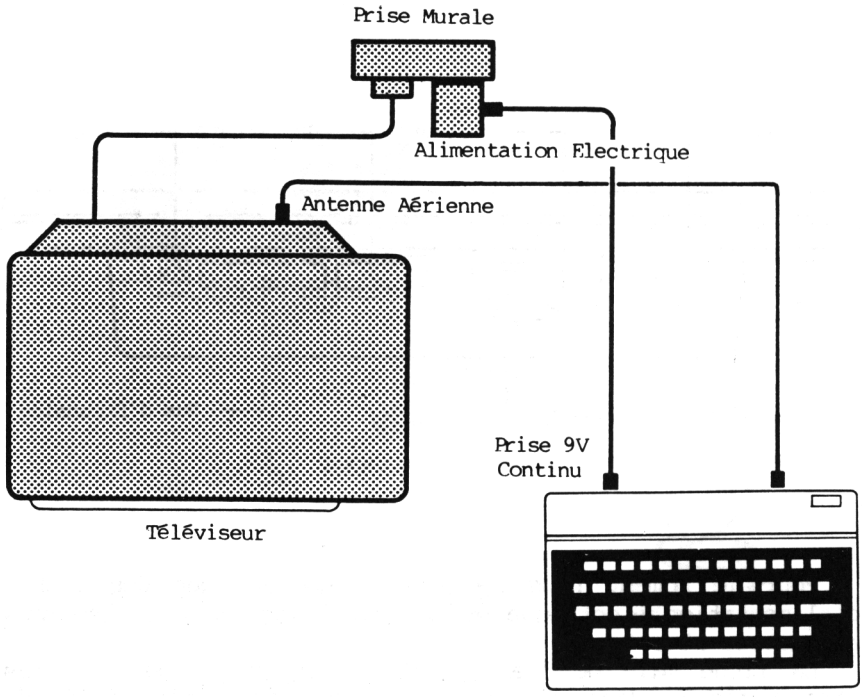


Pour le moment nous ne nous occuperons que des fiches d'alimentation électrique et de télévision. Nous reviendrons aux autres plus tard.

La fiche mâle de l'alimentation électrique ne doit pas être branchée dans la fiche d'alimentation électrique de l'ordinateur. Branchez d'abord l'alimentation électrique dans la prise murale. Si la prise murale a un interrupteur, alors positionnez-le .

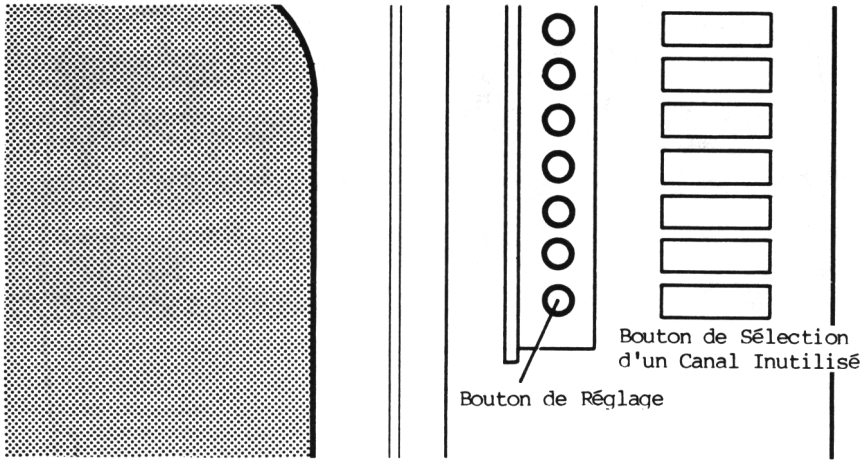
Débranchez la prise d'antenne aérienne de votre téléviseur et utilisez le cordon spécial fourni avec votre ORIC pour connecter la prise d'antenne de votre téléviseur sur la prise sortie TV de votre ordinateur. Si votre téléviseur a deux sorties UHF et VHF utilisez la sortie marquée UHF.

Si votre téléviseur a une antenne interne vous trouverez peut-être un interrupteur à côté de la prise d'antenne externe (normalement non utilisée). Cet interrupteur connecte sur l'antenne externe et déconnecte l'antenne interne. S'il existe, cet interrupteur doit être utilisé.



Branchez la télévision et branchez ensuite l'alimentation électrique de l'ordinateur. Baissez complètement le volume de la télévision.

Si votre appareil de télévision a des boutons de sélection de canal comme sur le diagramme ci-dessous, enfoncez le bouton d'un canal non utilisé habituellement.



Réglez votre téléviseur en tournant le bouton de contrôle jusqu'à ce que le message :

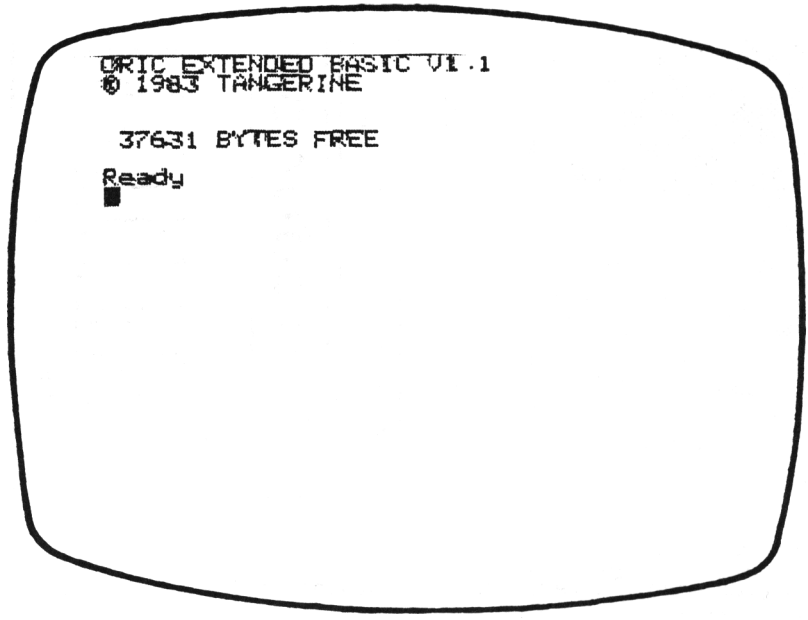
'ORIC EXTENDED BASIC V**
©1983 TANGERINE

***** BYTES FREE

Ready'

apparaisse en noir sur un écran blanc. Les nombres représentés par les étoiles varieront en fonction de la taille mémoire (16k ou 48k) et de la 'version' de la machine.

Le nombre d'octets libres (bytes free) est inférieur au total de la mémoire. Une partie de la mémoire est utilisée pour contrôler l'affichage à l'écran.

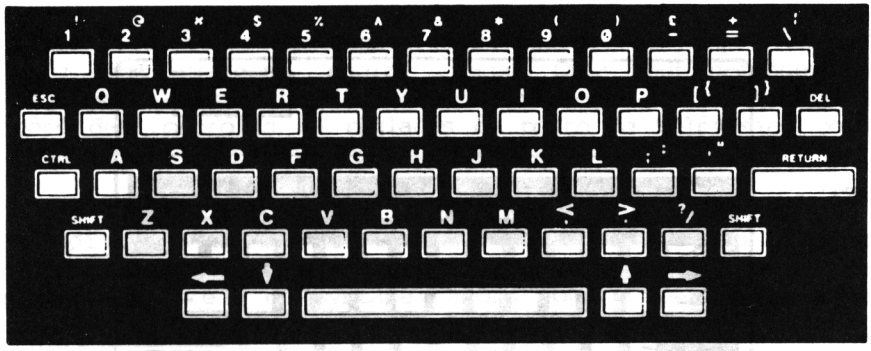


Si au lieu de ce message vous avez une image plutôt bizarre en noir et blanc sur votre écran, allumez et rééteignez l'ordinateur de nouveau. Faites ceci en débranchant et en reconnectant la prise à l'arrière de l'ordinateur. Elle est solide et ne s'usera pas. Allumer et éteindre l'ordinateur à l'aide de l'interrupteur mural fonctionne rarement car le courant revient trop doucement pour remettre correctement la machine en état.

En haut et à droite de l'écran, vous verrez le mot "CAPS" en lettres blanches sur fond noir. A gauche de l'écran, en dessous du mot "Ready" (Prêt) vous verrez un carré clignotant. Celui-ci s'appelle le "curseur".

COMMENCONS

Regardez le clavier.

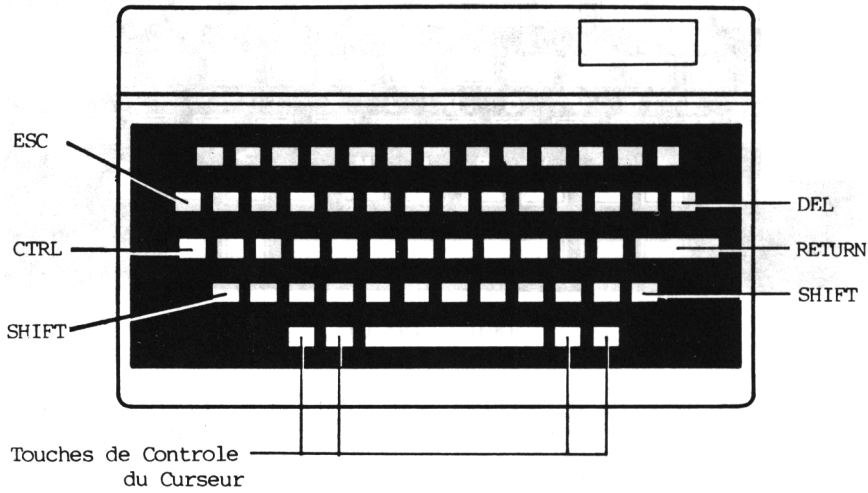


ORIC (V1.0)

Si vous avez déjà utilisé une machine à écrire, ne vous étonnez pas de la disposition du clavier de l'ORIC. Vous avez devant vous un clavier anglo-saxon "QWERTY" avec quelques touches supplémentaires (le clavier français est "AZERTY"). Il y a suffisamment d'espace entre les touches et les inscriptions y sont claires et aérées.

ORIC ATMOS (V1.1)

Notez la position des deux touches majuscules (SHIFT), de la touche contrôle (CTRL), de la touche échappatoire (escape, ESC), de la touche suppression (delete, DEL), des touches de contrôle du curseur et de la touche retour chariot (RETURN).



Voici quelques points importants à noter.

1. Je vais vous demander plus tard dans ce chapitre d'appuyer sur une touche SHIFT (l'une des deux touches SHIFT fera l'affaire, elles sont toutes deux pareilles pour l'ordinateur), et une autre touche simultanément. Ce que je veux réellement dire est que vous presserez la touche SHIFT d'abord et que vous la maintiendrez enfoncée pendant que vous appuierez sur l'autre touche ; alors vous les relâcherez toutes les deux. Beaucoup d'erreurs de frappe sont dues au fait que la touche SHIFT n'est pas enfoncée d'abord quand on demande une touche avec l'utilisation simultanée de la touche SHIFT.

2. Exactement de la même manière je vais vous demander d'enfoncer la touche CTRL et une autre touche simultanément. Dans ce cas la touche CTRL est enfoncée d'abord et maintenue enfoncée, tandis que l'autre touche est pressée, et ensuite les deux touches sont relâchées.

3. A partir de maintenant, quand je vous demanderai d'enfoncer une touche, je voudrais dire enfoncer et ensuite relâcher. Si je vous demande de maintenir une touche enfoncée je vous le dirai.

4. Si vous tapez un mauvais caractère, enfoncez la touche marquée DEL (juste au-dessus de la grande touche RETURN sur la droite du clavier). Cela annulera le dernier caractère tapé.

Tapez **PAPER 1** et appuyez sur la touche **RETURN.**

Si vous avez de la chance, le "papier" - c'est-à-dire la zone à l'intérieur de la bordure noire - se changera en rouge. Plus probablement elle se changera en une tonalité de gris. Si c'est le cas réglez votre téléviseur jusqu'à ce que vous ayez une zone "papier" rouge. Cela peut demander un réglage délicat. Si un interrupteur marqué AFT se trouve à côté des contrôles de réglages du poste de télévision cela vous aidera de fermer cet interrupteur.

Si au lieu de changer la couleur de la zone "papier", l'ordinateur imprime le message "?SYNTAX ERROR" sur l'écran (en jargon informatique, nous disons qu'il retourne un message d'erreur de syntaxe) alors vous n'avez pas frappé "**PAPER 1**" correctement. Dans ce cas, retapez-le.

Quand vous avez tapé la commande **PAPER 1**, vous n'avez pas eu besoin d'appuyer sur la touche SHIFT pour obtenir des lettres majuscules. Toutes les touches alphabétiques donnent des lettres majuscules lorsqu'elles sont enfoncées. C'est ce que CAPS dans le coin supérieur droit signifie. Vous n'utiliserez pas de lettres minuscules pendant encore un certain temps, ne vous en inquiétez donc pas.

Amusons-nous un peu avec l'ordinateur. Je n'expliquerai pas ce que je vais vous demander de faire à ce stade. Suivez mes indications. Tout vous sera expliqué plus tard.

ORIC LE BRUYANT

Tapez ZAP et appuyez sur la touche **RETURN.**

Certaines commandes ne sont pas si difficiles que cela à comprendre !

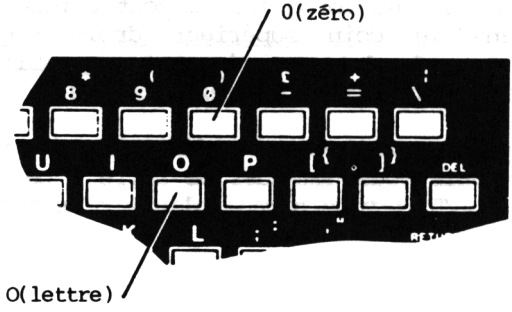
Maintenant essayez **PING** suivi de nouveau par la touche **RETURN.**

Tapez :

SHOOT:WAIT9:PING:WAIT9:ZAP:WAIT5:EXPLODE

(les deux points : sont obtenus en enfonçant simultanément la touche SHIFT et la touche à la droite de la touche L).

Faites bien attention à utiliser la lettre O dans SHOOT et EXPLODE et non le nombre zéro (0). Le nombre zéro (0) a une barre en travers sur le clavier de telle manière que vous puissiez voir la différence. De nombreux messages d'erreur de syntaxe sont dûs à la confusion entre ces deux touches.



Appuyez sur **RETURN.**

Vous devez avoir déjà découvert quelque chose à propos de l'ORIC. Il peut faire beaucoup de bruit - les faibles gazouillis produits par certains autres micro-ordinateurs ne sont pas pour les utilisateurs de l'ORIC !

Encore une explosion - et j'espère que vous avez des murs épais ou des voisins tolérants.

Tapez :

FOR N=1 TO 10:ZAP:WAIT5:NEXT

suivi comme d'habitude par la touche **RETURN.**

DE TOUTES LES COULEURS

Essayons quelque chose de plus calme.

Tapez :

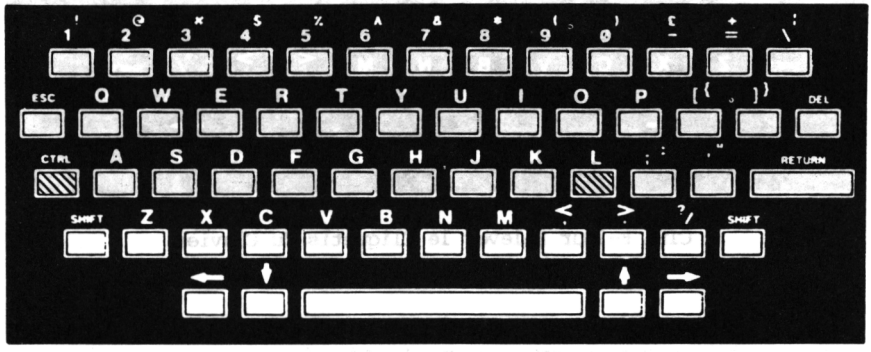
FOR N=0 TO 7:PAPER N:WAIT 100:NEXT

et appuyez sur RETURN.

De jolies couleurs n'est-ce-pas ?

NETTOYER L'ECRAN

Maintenant l'écran est plein de toutes sortes de choses. Voyons-voir comment on peut le nettoyer (l'effacer).
Appuyez sur la touche CTRL et la touche L simultanément.



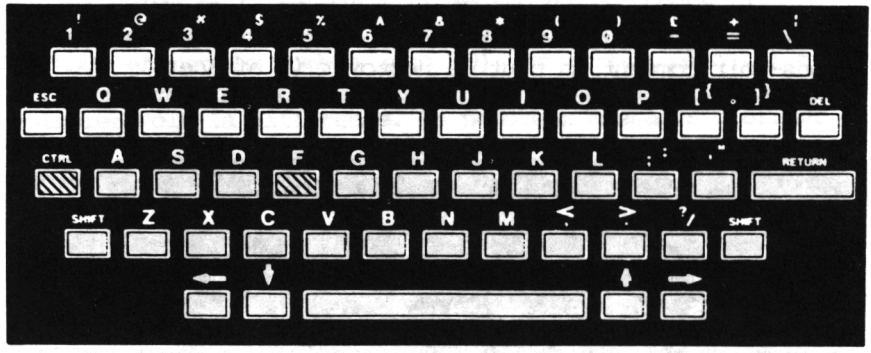
CTRL L-Pour Effacer l'Ecran

Et voici rapidement un écran parfaitement propre.

ENLEVER LE CLIQUETIS DES TOUCHES

Vous avez déjà remarqué que, quand vous appuyez sur une touche, cela produit un clic assez fort. C'est pour vous faire savoir que la touche a été enfoncée. Si vous aimez cette idée, très bien ! Si, comme moi, vous trouvez que le toucher du clavier est une information en retour amplement suffisante et que le cliquetis est une nuisance, alors vous voulez savoir comment vous en débarrasser.

Pour supprimer le cliquetis du clavier, enfoncez les touches CTRL et F simultanément.

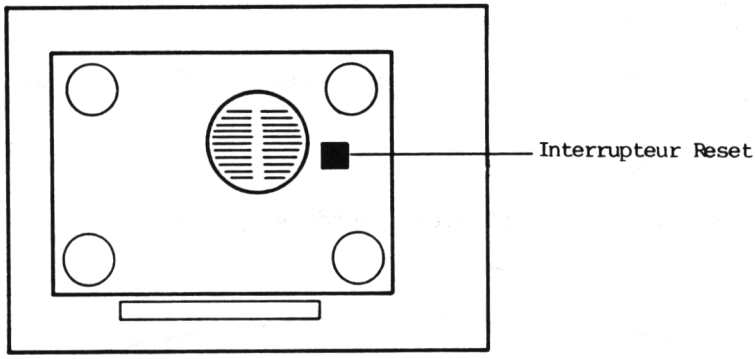


CTRL F-Pour Enlever le Cliquetis du Clavier

Si vous voulez obtenir de nouveau le cliquetis du clavier, enfoncez les touches CTRL et F simultanément.

L'INTERRUPTEUR RESET

Maintenant explorons un peu l'ORIC. Des choses très bizarres vont arriver. Aussi je ferai mieux de vous parler du bouton de secours. Si vous soulevez l'ORIC gentiment, de telle manière que vous n'enleviez pas les connections de l'arrière de la machine, vous verrez un petit trou carré. A l'intérieur de ce trou se trouve un interrupteur. C'est l'interrupteur RESET de l'ORIC. Vous aurez besoin d'un crayon ou d'un tournevis pour faire fonctionner cet interrupteur. Quand vous le faites l'écran se vide, et le message 'Ready' apparaît.



L'avantage de l'interrupteur RESET est qu'il permet un "démarrage à chaud". Cela veut dire que les "programmes" (voir chapitre 2) ne sont pas perdus, comme ils le seraient si vous coupiez ou remettiez l'alimentation électrique. Si vous êtes perdu et que n'arriviez pas à ramener la machine à un mode de fonctionnement normal, il est réconfortant de savoir que l'interrupteur RESET est là, mais n'essayez pas de l'utiliser trop souvent.

QUELQUES TRUCS AMUSANTS

Tapez **PAPER0:INK3**

et appuyez sur **RETURN**
Appuyez sur **CTRL** et **L** simultanément.

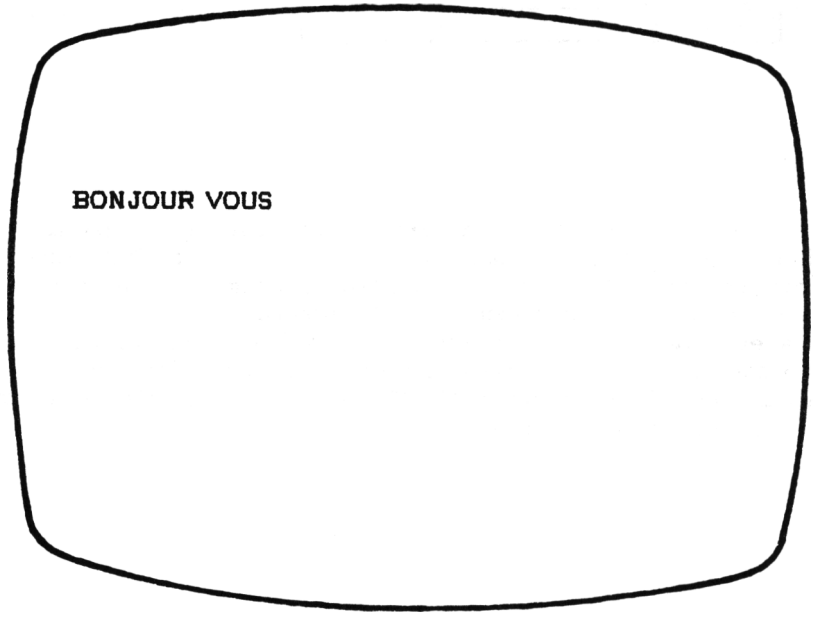
Vous allez obtenir un écran noir, vide, excepté le curseur clignotant dans le coin supérieur gauche.

Appuyez sur **RETURN** pour déplacer le curseur d'une ligne vers le bas.
Appuyez sur **CTRL** et **D** simultanément.
Appuyez sur **ESC** (touche ESCAPE).
Appuyez sur la touche **J**.

Jusqu'à maintenant rien de sensationnel n'est arrivé. Toutefois, si vous tapez :

BONJOUR VOUS

Vous trouverez le résultat plutôt surprenant.



Appuyez sur **RETURN**. Ignorez le message bizarre que cela produit.

Appuyez sur **CTRL** et **L** simultanément pour nettoyer et vider l'écran et appuyez sur **RETURN** pour descendre le curseur d'une ligne.

Appuyez sur **ESC** suivi par **N**.

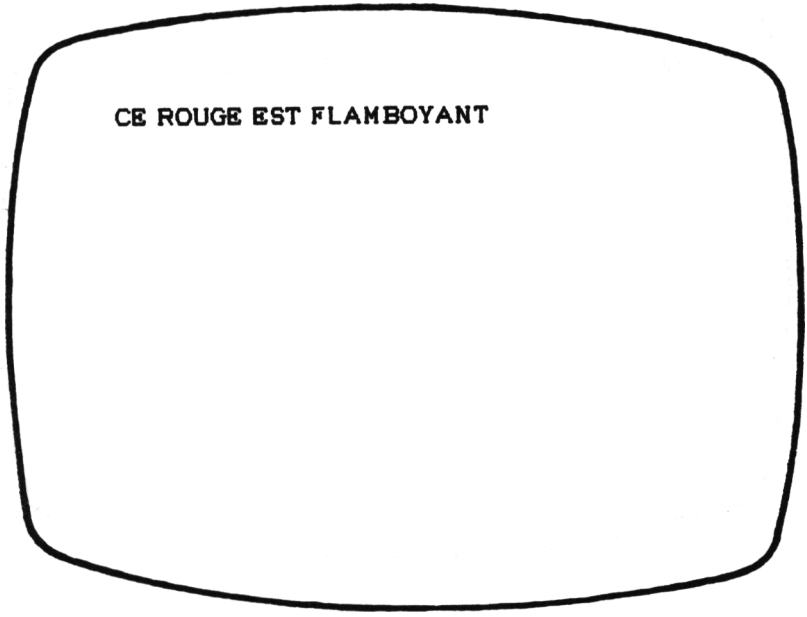
Tapez :

CE ROUGE EST

Appuyez sur **ESC** suivi par **A**.

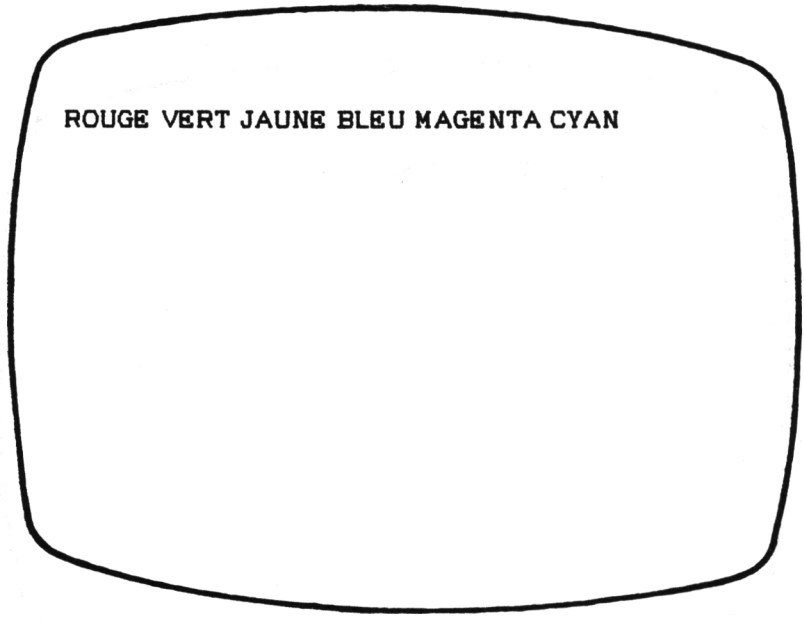
Tapez **FLAMBOYANT**

Comment trouvez-vous cela !



Appuyez sur **RETURN** en ignorant le message bizarre que cela produit. Videz l'écran en appuyant sur les touches **CTRL** et **L** simultanément et appuyez sur la touche **RETURN** pour descendre le curseur d'une ligne.

Appuyez sur **ESC** suivi par **J**.
Appuyez sur **ESC** suivi par **A** et tapez **ROUGE**.
Appuyez sur **ESC** suivi par **B** et tapez **VERT**.
Appuyez sur **ESC** suivi par **C** et tapez **JAUNE**.
Appuyez sur **ESC** suivi par **D** et tapez **BLEU**.
Appuyez sur **ESC** suivi par **E** et tapez **MAGENTA**.
Appuyez sur **ESC** suivi par **F** et tapez **CYAN**.



Avant de quitter les lettres majuscules, essayons encore une démonstration. Comme précédemment appuyez sur **RETURN**, et ignorez le message bizarre. Appuyez sur **CTRL** et **L** simultanément pour nettoyer et vider l'écran, suivi par **RETURN** pour déplacer le curseur d'une ligne vers le bas.

Appuyez sur **ESC** suivi par **W**;

Appuyez sur **ESC** suivi par **@** (@ est produit par l'enfoncement simultané des touches SHIFT et 2);

Appuyez sur **ESC** suivi par **J**;

et tapez **NOIR SUR BLANC.**

Déplacez le curseur de deux lignes vers le bas, en utilisant la touche de contrôle du curseur avec la flèche pointant vers le bas qui se trouve à gauche de la barre d'espace.
Placez le curseur à l'extrême gauche de l'écran, en utilisant la touche de contrôle du curseur avec la flèche pointant vers la gauche.

Appuyez sur **ESC** suivi par **V**;
ESC suivi par **A**;
ESC suivi par **J**;
et tapez **ROUGE SUR CYAN.**

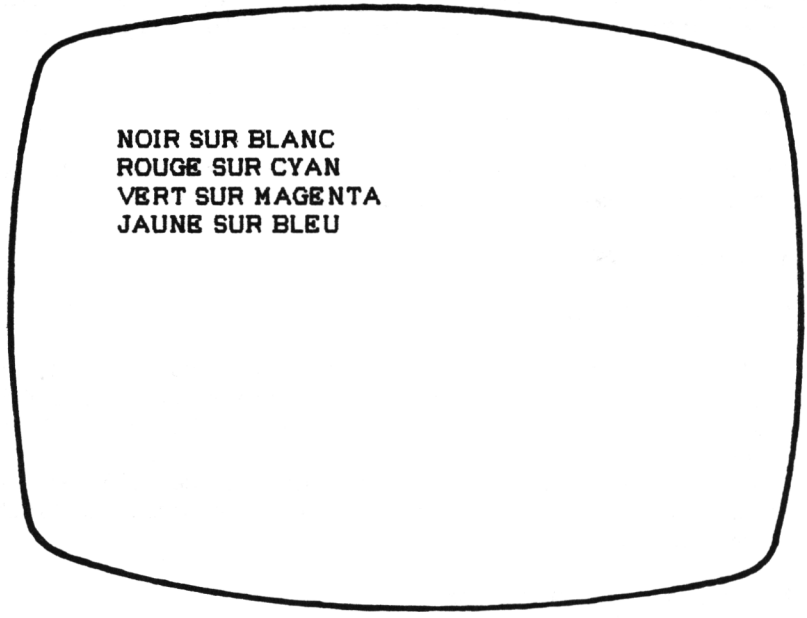
Comme précédemment, utilisez les touches de contrôle du curseur pour le déplacer vers le bas et vers la gauche au début de la ligne.

Appuyez sur **ESC** suivi par **U**;
ESC suivi par **B**;
ESC suivi par **J**;
et tapez **VERT SUR MAGENTA.**

Utilisez les touches de contrôle du curseur pour positionner de nouveau le curseur au début de la double ligne suivante. Vous devriez être habitués à faire cela maintenant.

Appuyez sur **ESC** suivi par **T**;
ESC suivi par **C**;
ESC suivi par **J**;
et tapez **JAUNE SUR BLEU.**

Finalement appuyez sur **CTRL** et **Q** ensemble pour cacher le curseur. J'espère que vous serez d'accord pour dire que c'est une bonne démonstration des possibilités d'entrée au clavier de cette machine.


NOIR SUR BLANC
ROUGE SUR CYAN
VERT SUR MAGENTA
JAUNE SUR BLEU

QUELQUES BELLES IMAGES

Les mots **sont** ennuyeux, **voyons voir quelques images !**

La première chose à faire est de récupérer le curseur. Il est difficile de travailler sans lui.

Appuyez sur **CTRL** et **Q** simultanément.

Appuyez sur **RETURN** et appuyez sur **CTRL** et **L** simultanément; alors appuyez de nouveau sur **RETURN.**

Appuyez sur **ESC** suivi par **K**

et tapez

NS£S£S£S£,W[^/^

(^ est produit en enfonçant simultanément les touches SHIFT et 6).

Utilisez les touches de contrôle du curseur pour déplacer le curseur en bas et à gauche au début de la première ligne blanche (comme vous l'avez fait dans la dernière démonstration).

Appuyez sur **CTRL** et **D** simultanément et ensuite sur **CTRL** et **T** simultanément.

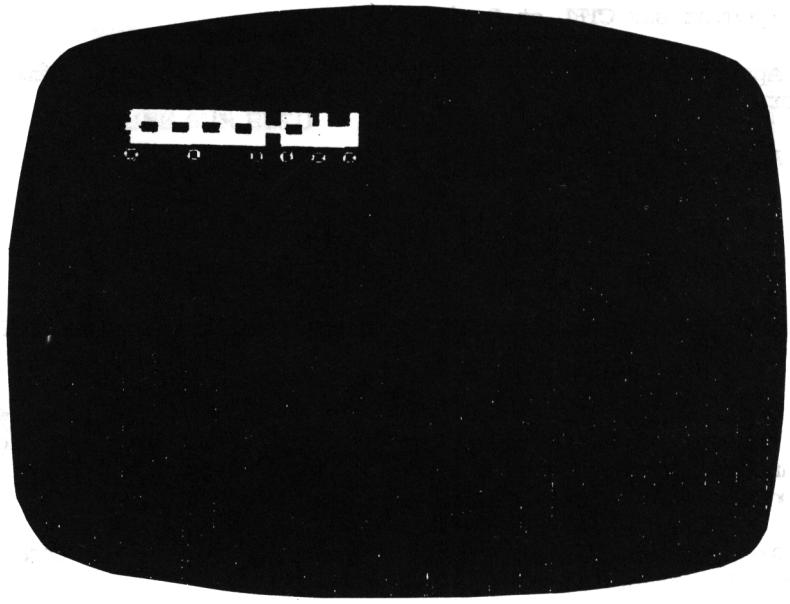
Appuyez sur la touche O (c'est-à-dire sur la lettre O et non zéro).

Appuyez sur la **barre d'espacement** trois fois.

Appuyez sur la touche O et sur la **barre d'espacement** trois fois.

Appuyez sur O **espace** O **espace** O **espace** O.

Appuyez sur **CTRL** et **Q** simultanément pour cacher le curseur.



Appuyez sur **CTRL** et **T** simultanément, ensuite sur **CTRL** et **Q** simultanément.

Appuyez sur **RETURN** et nettoyez l'écran en enfonçant les touches **CTRL** et **L** simultanément.

Tapez **HIRES** et appuyez sur **RETURN**.

Le curseur devrait se déplacer jusqu'au bas de l'écran. Vous devriez voir le mot "Ready" en lettres jaunes près du curseur.

Tapez : **CURSET 125,99,3:CIRCLE 90,1**

et appuyez sur **RETURN**.

Un cercle devrait apparaître sur l'écran.

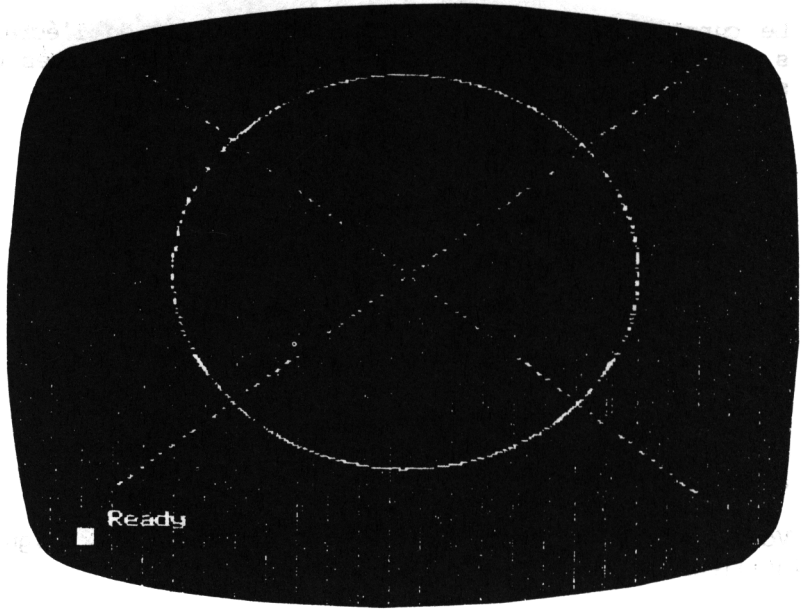
Tapez : **CURSET 12,0,3:PATTERN 51:DRAW 227,199,1**

et appuyez sur **RETURN**.

Tapez : **CURSET 12,199,3:DRAW 227;-199,1**

et appuyez sur **RETURN**.

Vous devriez maintenant avoir un cercle et deux lignes pointillées qui se croisent.


Ready

Pas mal pour un tout petit ordinateur, pensez-vous, mais ORIC a bien d'autres choses encore à votre disposition.

Ce que vous allez faire maintenant dépend de la version de la machine que vous avez.

Si vous avez la version 48K de l'ORIC, tapez :

```
FOR N = #A000 to #BFE0 STEP 40:POKE N,
INT(RND(1)*7+1):NEXT
```

et appuyez sur **RETURN**.

Si vous avez la version 16k de l'ORIC, tapez :

```
FOR N = #2000 TO #3FE0 STEP 40:POKE N,
INT(RND(1)*7+1):NEXT
```

et appuyez sur **RETURN**.

Attention : dans l'un ou l'autre cas, appuyez sur **RETURN** après **NEXT**, **pas** après **N,**.

J'espère que ces démonstrations vous ont convaincu que vous aviez une petite machine remarquable et versatile. Vous pouvez voir quelques régularités dans ce que nous avons déjà fait (sinon ne vous inquiétez pas) et vous êtes peut-être intéressé par ce qui aurait pu arriver si vous aviez changé quelque peu les instructions.

Allez-y et essayez !

Chapitre 2
LES TOUCHES DU SUCCES

LE CLAVIER

Dans le chapitre précédent vous avez utilisé le clavier pendant un certain temps et vous commencez probablement à le maîtriser. L'espacement entre les touches est le même que sur un clavier standard de machine à écrire.

Appuyez sur l'interrupteur RESET sous la machine ou encore éteignez et rallumez l'ordinateur, cela remet la machine à son état initial.

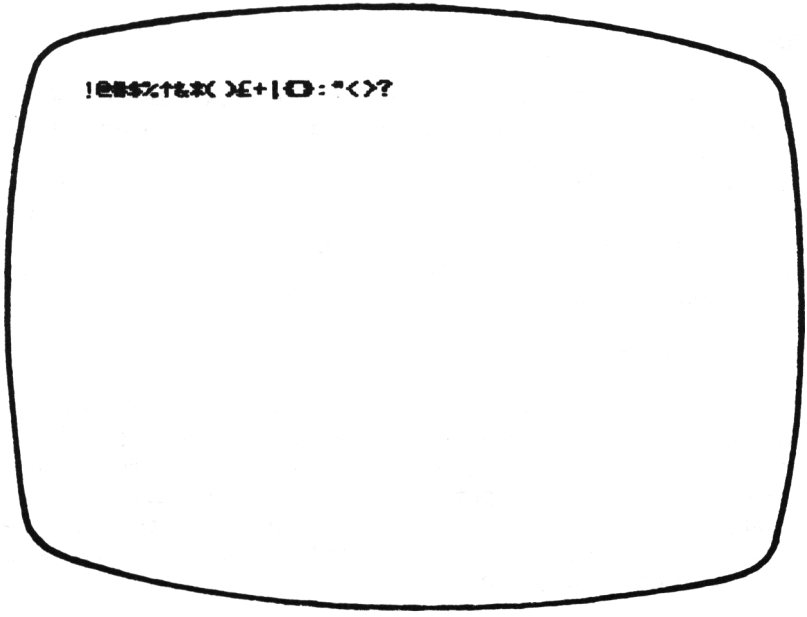
TOUCHES "SHIFTEES"

Quelques-unes des touches ont deux caractères écrits dessus. En enfonçant sur cette touche le caractère du bas apparaît sur l'écran. En enfonçant la même touche et la touche SHIFT simultanément, (l'une des deux touches SHIFT fera l'affaire), alors le caractère supérieur apparaît.

Par exemple, en enfonçant la touche 3 on fait apparaître 3 sur l'écran. En enfonçant les touches 3 et SHIFT simultanément, on fait apparaître le symbole # sur l'écran (on dit que le symbole # est le 3 "shifté").

Pour utiliser ces touches - tapez :

```
!@#$%^&*()£+¦{} /:"<>?
```



Le symbole ^(6 shifté) est affiché comme ↑ sur l'écran.

TOUCHE D'EFFACEMENT

La touche d'effacement DEL est utilisée pour supprimer les entrées au clavier si, par exemple, vous tapez la mauvaise touche ou si vous changez d'avis.

DEL déplace le curseur d'un espace sur la gauche et supprime le dernier caractère entré à la gauche du curseur. Si le curseur est à l'extrêmité gauche d'une ligne alors DEL se déplacera à l'extrêmité droite de la ligne précédente assumant par là qu'il y a un caractère dans cette position qui fait partie de ce qui est entré en ce moment au terminal. Sinon DEL n'a pas d'effet.

Si vous essayez d'effacer des entrées qui ont été terminées (c'est-à-dire que vous avez déjà appuyé sur la touche RETURN) vous obtiendrez des résultats bizarres.

RETURN

La touche RETURN est utilisée pour terminer les entrées au terminal. Dans le chapitre précédent vous avez vu que l'ordinateur obéit à une commande (il a fait ce que vous lui avez dit de faire) seulement après que RETURN ait été enfoncé. Par exemple, quand vous avez tapé "ZAP" rien n'est arrivé jusqu'à ce que vous ayez appuyé sur RETURN.

RETURN est aussi utilisé pour entrer des "programmes" d'ordinateur ce que nous verrons plus tard dans ce chapitre.

REPETITION AUTOMATIQUE

Appuyez sur une touche, disons la touche Q, et gardez-la enfoncée jusqu'à ce que vous ayez toute une ligne de Q. Maintenir une touche enfoncée a le même effet que de l'enfoncer de manière répétée. C'est un dispositif très utile si vous souhaitez taper par exemple un tas d'espaces. Cela fonctionne également avec la touche DEL, ce qui permet d'effacer toute une ligne.

TAMPON DU CLAVIER

Enfoncez une touche alphabétique ou numérique et maintenez-la enfoncée. Le caractère choisi apparaîtra sur l'écran de manière répétée jusqu'à ce que vous entendiez éventuellement un "ping". Quand vous entendrez le ping, relâchez la touche.

Vous êtes autorisé à taper seulement un nombre maximum de caractères avant d'enfoncer la touche RETURN. Quand vous tapez les caractères, ils sont placés dans un endroit appelé le **tampon du clavier**, où ils sont stockés jusqu'à ce que la touche **RETURN** soit enfoncée. Le tampon du clavier peut contenir un nombre limité de caractères. Le ping est la méthode employée par l'ORIC pour vous prévenir que le tampon du clavier est rempli.

LETTRES MINUSCULES

Le clavier de l'ORIC comme la plupart des claviers d'ordinateur est similaire dans son fonctionnement à ceux des machines à écrire standards. Toutefois, jusqu'à présent il a fonctionné comme un clavier avec la touche "capitals lock" enfoncée. Les touches alphabétiques ont donné des lettres capitales lorsqu'elles étaient enfoncées et la touche SHIFT ne les a pas modifiées.

Pour enlever capitals lock appuyez sur CTRL et T simultanément.

Le message CAPS dans un coin supérieur droit de l'écran disparaît et lorsque l'on tape une touche alphabétique on obtient un caractère minuscule ; en enfonçant la même touche simultanément avec la touche SHIFT on obtient la même lettre en majuscule.

Vous travaillerez avec CAPITALS LOCK NON POSITIONNÉ si par exemple vous utilisez l'ORIC pour taper une lettre. Toutefois, normalement vous travaillerez avec capitals lock positionné car toutes les instructions que vous donnerez à l'ordinateur doivent être en lettres majuscules.

Pour remettre capitals lock en position appuyez de nouveau sur CTRL et T simultanément.

CARACTERES DE CONTROLE

Dans le chapitre précédent, et ci-dessus, nous avons utilisé la touche CTRL, enfoncée simultanément avec une autre touche pour faire faire à l'ordinateur des choses aussi variées que nettoyer l'écran, positionner et dépositionner le cliquetis du clavier, positionner et dépositionner capitals lock. Enfoncer la touche CTRL simultanément avec une autre touche pro-duit ce qui est appelé un **caractère de contrôle.** Nous étudierons à nouveau les caractères de contrôle au chapitre 7.

Dans le reste de ce livre, j'indiquerai l'enfoncement simultané de CTRL et d'autres touches sous la forme **CTRL A, CTRL C,** etc... **CTRL A** signifie que les touches CTRL et A sont enfoncées simultanément.

Je vais commencer par lister les caractères de contrôle avec une brève explication en référence. Quand cela sera nécessaire, je donnerai un peu plus de détails. Ne vous inquiétez pas si vous ne comprenez pas toutes les explications tout de suite. Nous les étudierons à nouveau dans les chapitres suivants.

CTRL A - copie des caractères de l'écran dans le tampon du clavier.

CTRL C - arrête une boucle du programme ou le défilement du clavier.

CTRL D - provoque une double impression (bascule à positionner ou à dépositionner).

CTRL F - contrôle du cliquetis du clavier (bascule à positionner ou à dépositionner).

CTRL G - provoque un "ping".

CTRL H - déplace le curseur vers la gauche.

CTRL I - déplace le curseur vers la droite.

CTRL J - déplace le curseur vers le bas.

CTRL K - déplace le curseur vers le haut.

CTRL L - vide l'écran.

CTRL M - fonctionne comme la touche RETURN.

CTRL N - cache une ligne.

CTRL O - cache toutes sorties à venir à l'écran (bascule à positionner ou à dépositionner).

CTRL P - contrôle l'imprimante (bascule à positionner ou à dépositionner).

CTRL Q - cache le curseur (bascule à positionner ou à dépositionner).

CTRL S - cache la sortie à l'écran (fonctionne comme CTRL O).

CTRL T - contrôle capitals lock (bascule à positionner ou à dépositionner).

CTRL X - enlève une ligne de programme de la mémoire.

CTRL Z
CTRL [- affecte les couleurs d'arrière-plan et d'avant-plan pour le reste de la ligne.

CTRL] - permet l'impression dans la colonne de gauche la plus éloignée.

CTRL A

Ce caractère de contrôle est utilisé dans la procédure d'**édition**. C'est-à-dire lorsque l'on change le contenu de la "mémoire" de l'ordinateur. Nous étudierons cette utilisation en détail plus tard dans ce chapitre.

CTRL C

C'est le caractère "d'arrêt" ou de "rupture" pour l'ORIC. Vous le verrez en fonctionnement tout au long de ce livre.

CTRL D

Cela cause une double impression. Pour voir son fonctionnement tapez **CTRL D** et ensuite tapez n'importe quel message. **CTRL D** est ce qui est connu sous le nom de "bascule". Cela signifie que si la double impression est dépositionnée, **CTRL D** la repositionne et que si la double impression est positionnée, **CTRL D** la dépositionne.

Nous ne voulons pas de double impression maintenant; aussi tapez **CTRL D** de nouveau pour le dépositionner.

CTRL F, CTRL G, CTRL L

Vous avez utilisé **CTRL F** ET **CTRL L** dans le chapitre précédent et ils doivent vous être familiers. **CTRL F** est une clé à bascule. La description de **CTRL G** va de soi, je l'espère.

CTRL H, CTRL I, CTRL J, CTRL K, CTRL M

Ces caractères de contrôle sont rarement utilisés comme entrées directes au clavier. Il est plus facile d'utiliser les touches de contrôle du curseur et la touche RETURN qui assurent les mêmes fonctions. L'utilisation des caractères de contrôle qui ne sont pas produits par des entrées au clavier est traité dans le chapitre 7.

CTRL N

Si vous l'utilisez après avoir tapé une commande, alors cette commande disparaîtra de l'écran. Toutefois la commande sera tout de même exécutée si vous appuyez sur RETURN.

CTRL O

Bascule qui supprime ou au contraire redonne l'information qui serait normalement affichée sur l'écran.

CTRL S

Si vous positionnez **CTRL S** les caractères rentrés au clavier ne sont pas affichés à l'écran. Un des usages possibles est pour les messages "secrets" ou pour les applications où les entrées au clavier pourraient altérer un affichage à l'écran. **CTRL S** est une bascule.

CTRL Q

Vous avez utilisé **CTRL Q** au chapitre 1. Le curseur clignotant peut perturber les affichages à l'écran et il est utile de pouvoir être capable de s'en débarrasser. **CTRL Q** est une bascule.

CTRL T

Nous avons déjà traité CTRL dans ce chapitre. **CTRL T** est une bascule.

CTRL X

Cette clé annule les entrées au clavier faites à la gauche de ce caractère. Il peut être utile lorsque l'on entre des "programmes". Voir plus loin dans ce chapitre.

CTRL Z, CTRL [

Ces caractères de contrôle peuvent tous deux affecter les couleurs, d'arrière-plan et d'avant-plan des lignes imprimées sur l'écran. Ils peuvent avoir toute une gamme d'autres possibilités plutôt bizarres, l'effet dépendant de la touche qui est pressée après que ce caractère de contrôle ait été tapé. Ce que ces caractères de contrôle font est de préparer les attributs de la machine, attributs que nous étudierons plus tard. **CTRL Z** et **CTRL [** ne sont pas particulièrement utiles mais il est intéressant de jouer avec eux et de voir les effets bizarres qu'ils peuvent produire.

CTRL]

Nous verrons ce caractère de contrôle après avoir étudié les attributs.

ATTRIBUTS

Si certaines des choses que nous pouvons faire avec la touche **CTRL** semblent bizarres et magnifiques, elles sont toutefois insignifiantes à côté des effets plus que bizarres que nous avons vu lorsque nous avons utilisé la touche **ESC.**

L'effet de la touche ESC est déterminé par la touche enfoncée immédiatement après celle-ci. La touche ESC prépare l'ordinateur pour mettre quelque chose appelé un **attribut** dans le premier caractère libre sur l'écran. La touche suivant le caractère **ESC** détermine quel est l'attribut placé dans cet espace.

L'espace sur l'écran dans lequel l'attribut est entré est laissé blanc. L'attribut détermine une caractéristique de tous les caractères imprimés à sa droite sur l'écran jusqu'à la fin de la ligne ou jusqu'à ce qu'un autre attribut soit rencontré. Il y a un certain nombre de méthodes pour mettre les attributs sur l'écran (ou pour être plus exact dans la section de la mémoire de l'ordinateur qui contrôle l'écran). Ici, toutefois, nous ne considérons que l'entrée directe au clavier et nous regarderons donc les entrées commandées par la touche **ESC.**

La touche **ESC** n'a pas d'effet en soi. Ses effets sont déterminés par la touche qui est pressée ensuite (comparée à la touche **CTRL** qui elle nécessite que l'on enfonce une touche simultanément avec elle). Quand je parle de **ESC** et d'une autre touche (disons par exemple **ESC** et **A**) je veux dire la touche **ESC** suivie par la touche mentionnée.

Je résumerai d'abord les effets des entrées de la touche **ESC** et ensuite je développerai ces résumés si nécessaire.

ESC et @ - avant-plan noir (encre).

ESC et A - avant-plan rouge.

ESC et B - avant-plan vert.

ESC et C - avant-plan jaune.

ESC et D - avant-plan bleu.

ESC et E - avant-plan magenta (pourpre).

ESC et F - avant-plan cyan (bleu clair).

ESC et G - avant-plan blanc.

ESC et H - caractère standard, simple hauteur, permanent.

ESC et I - caractère alternatif, simple hauteur, permanent.

ESC et J - caractère standard, double hauteur, permanent.

ESC et K - caractère alternatif, double hauteur, permanent.

ESC et L - caractère standard, simple hauteur, clignotant.

ESC et M - caractère alternatif, simple hauteur, clignotant.

ESC et N - caractère standard, double hauteur, clignotant.

ESC et O - caractère alternatif, double hauteur, clignotant.

ESC et P - arrière-plan noir (papier).

ESC et Q - arrière-plan rouge.

ESC et R - arrière-plan vert.

ESC et S - arrière-plan jaune.

ESC et T - arrière-plan bleu.

ESC et U - arrière-plan magenta (pourpre).

ESC et V - arrière-plan cyan (bleu clair).

ESC et W - arrière-plan blanc.

AVANT-PLAN ET ARRIERE-PLAN

Ces termes sont (je l'espère) suffisamment explicatifs. Vous pouvez utiliser les attributs de l'avant-plan et de l'arrière-plan au début d'une colonne pour positionner les coloris de l'encre et du papier, et au milieu d'une colonne pour changer ces couleurs.

CTRL]

Cet attribut prend normalement une position de caractère. Aussi lorsque nous entrons des textes normalement les deux premiers caractères de chaque ligne sont réservés pour les attributs. Si vous souhaitez imprimer dans les deux caractères des colonnes de l'extrême gauche sur l'écran, **CTRL]** vous permet de faire cela. **CTRL]** est une bascule.

CLIGNOTANT ET PERMANENT

Si une ligne de texte est précédée par un attribut clignotant il clignotera. C'est-à-dire qu'il changera de la couleur d'arrière-plan à la couleur d'avant-plan à peu près trois fois par seconde. Le texte non précédé par un attribut clignotant restera permanent en couleur d'avant-plan. **ESC** suivi par **L,M,N** et **Q** insère des attributs clignotants.

CARACTERES STANDARDS ET CARACTERES ALTERNATIFS

En plus des caractères normaux, (A,B,C, ... 1,2,3, etc) ORIC a aussi une police de caractères alternatifs. C'est une collection de formes qui peuvent être utilisées pour former des images telles par exemple le train que vous avez tapé dans le chapitre 1. Nous étudierons cet ensemble de caractères en détail au chapitre 12.

ESC suivi par **I,K,M** and **O** permet d'insérer des attributs de caractères alternatifs.

CARACTERES DOUBLE HAUTEUR

L'effet des attributs double hauteur varie en fonction de la ligne de l'écran dans laquelle elles sont utilisées. Si un attribut double hauteur par exemple est mis dans la première ligne du haut de l'écran (la ligne 0) vous obtiendrez la partie de la moitié inférieure des caractères en double hauteur que vous tapez.

Par exemple si vous tapez "A", vous obtiendrez "n", si vous tapez "C" vous obtiendrez "L".

Au contraire si l'attribut double hauteur est sur la ligne suivante (ligne 1) vous obtiendrez la moitié supérieure du caractère en double hauteur que vous tapez.

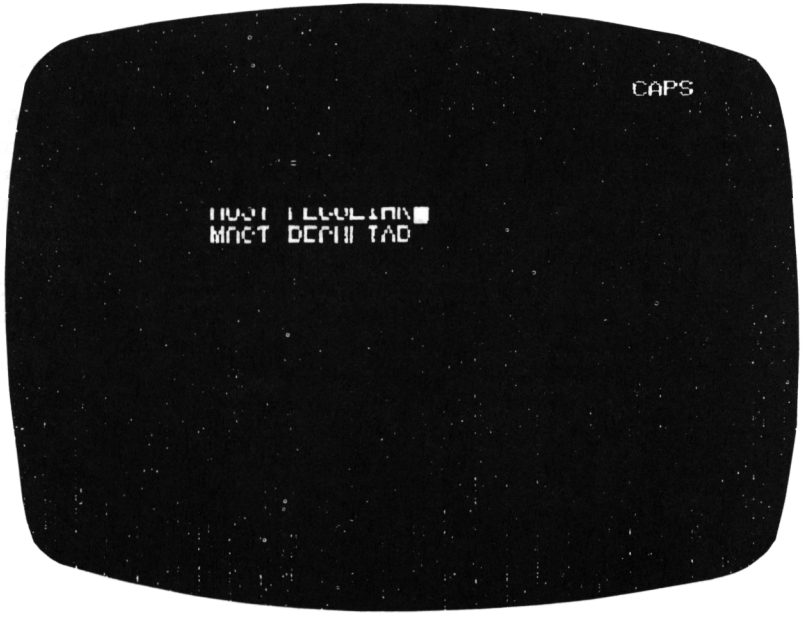
Par exemple si vous tapez "A" vous obtiendrez "^", si vous tapez "C" vous obtiendrez "Γ".

Cela continue ainsi jusqu'au bas de l'écran. Vous obtiendrez les moitiés inférieures sur les lignes 0,2,4 etc... (lignes paires) et les moitiés supérieures sur les lignes 1,3,5 etc... (lignes impaires).

Ainsi pour obtenir un message en double hauteur vous tapez d'abord le message sur une ligne impaire avec un attribut double hauteur et ensuite vous tapez le même message sur la ligne suivante à nouveau avec un attribut double hauteur. La moitié supérieure du message sur la ligne supérieure rejoint alors la moitié inférieure du message sur la ligne inférieure pour donner un message en double hauteur.

Taper un message deux fois est toutefois un peu difficile. Il doit y avoir, vous le pensez, un moyen plus facile. Oui, souvenez-vous de l'effet plutôt bizarre de **CTRL D.** Le message tapé avec ce caractère de contrôle apparaît deux fois sur deux lignes successives, juste ce que vous voulez pour une impression en double hauteur. Aussi pour obtenir une impression en double hauteur, positionnez **CTRL D** puis insérez un attribut double hauteur en utilisant **ESC** suivi par **J** (caractère standard statique) ou **K** (caractère alternatif statique) ou **N** (caractère standard clignotant) ou O (caractère alternatif clignotant). Quand le caractère de double hauteur est positionné l'attribut contrôlera les deux lignes. Les attributs double hauteur produisent aussi un arrière-plan noir. Aussi, soyez sûr que vous avez positionné une couleur d'avant-plan autre que noire, sinon vous ne verrez pas le message.

Faites bien attention à commencer par une ligne numérotée impaire. Sinon l'effet sera très particulier.



ATTRIBUTS FONCTIONNANT AVEC 60Hz

Quelques attributs chambouleront votre image sur l'écran car ils sont conçus pour fonctionner avec des appareils de télévision américains fonctionnant sur 60Hz. Evitez de faire suivre **ESC** par **X, Y, ^, £, /, ¦ ,],** ou } .

Chapitre 3
ENTREZ DANS L'ORIC

ENTREE DE PROGRAMMES

Aux chapitres 1 et 2 vous avez "tapé" quelques "commandes". En français courant cela veut dire que vous avez appuyé sur quelques touches et que cela fait faire quelque chose à l'ordinateur.

Une fois que l'ordinateur a fait ce que vous lui avez dit de faire, il a toutefois oublié la commande. Pour obtenir de l'ordinateur qu'il répète l'action, vous devriez retaper à nouveau toutes les touches de la même manière.

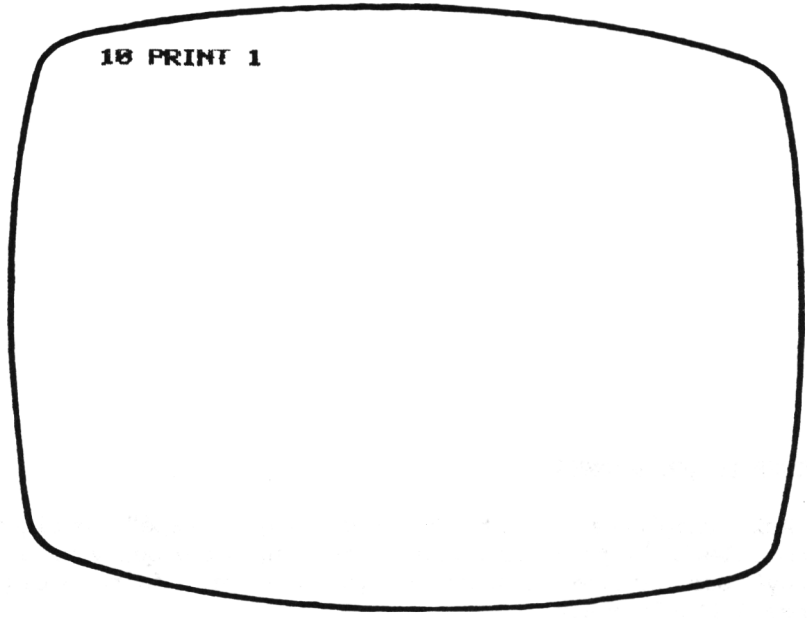
Une commande que l'ordinateur exécute immédiatement et qu'il oublie est appelée une commande "immédiate".

Essayons quelque chose de différent ; d'abord effacez l'écran en utilisant CTRL L, puis, tapez :

10PRINT1

et appuyez sur la touche RETURN. Vous n'avez pas insérer un espace avant et après PRINT bien que cela ne fasse pas de différence si vous le faites.

L'écran ressemble maintenant à cela :



Tapez RUN et appuyez sur la touche RETURN - le numéro "1" apparaît sur l'écran. Si au lieu de cela vous obtenez le message "SYNTAX ERROR" recommencez toute la procédure (en commençant en haut de la page).

Maintenant voici l'astuce - tapez RUN suivi de nouveau par la touche RETURN.

Et à nouveau vous obtenez le chiffre 1. L'ordinateur s'est souvenu de l'instruction et continuera à faire de même jusqu'à ce que vous supprimiez ou changiez quelque chose ou que vous coupiez l'alimentation électrique.

En numérotant l'instruction vous avez fait en sorte que la machine l'a stocké dans sa mémoire. De ce fait l'instruction n'est plus une commande immédiate mais maintenant une commande appartenant à un programme. Les programmes sont composés d'instructions avec des numéros de ligne. L'ordinateur stocke ces instructions et les exécute (les mène à bien) chaque fois que le programme est soumis à la commande RUN.

Vous avez juste entré et fait exécuté votre premier programme.

NUMEROS DE LIGNE

Les nombres au début des lignes de programmes, non seulement définissent que les instructions dans ces lignes sont des instructions de programme, mais ils sont aussi un des facteurs déterminant l'ordre dans lequel ces instructions seront exécutées.

Vous avez déjà la ligne :

10 PRINT 1

Tapez :

5 PRINT 3

et appuyez sur **RETURN.**

Tapez **RUN** suivi par **RETURN** pour faire exécuter le programme.

Vous verrez que l'écran montre maintenant les résultats suivants :

3
1

La ligne 5 est exécutée avant la ligne 10 bien qu'elle ait été rentrée après.

Tapez LIST suivi de la touche RETURN.

Les deux lignes que vous avez entrées vont apparaître maintenant à l'écran :

5 PRINT 3
10 PRINT 1

C'est ce qui est connu sous le nom de "listing de programme". Vous voyez que, non seulement la ligne 5 est "exécutée" avant la ligne 10, mais qu'elle est aussi listée avant elle.

ENTREE DE PROGRAMMES

Quand vous tapez un programme il y a quelques points à noter. Vous avez remarqué que vous n'avez pas mis d'espace entre le numéro de la ligne et l'instruction qui la suit. Quand vous avez listé le programme ORIC a mis les espaces pour vous.

Quand vous avez fini de taper une ligne vous avez appuyé sur la touche RETURN. Cela "termine" la ligne et "l'entre" en "mémoire" de manière à ce que la machine s'en souvienne et exécute les instructions quand la commande "RUN" sera donnée. Souvenez-vous d'appuyer sur la touche RETURN après avoir tapé chaque ligne de programme indiquée dans ce livre.

Si vous faites une erreur pendant que vous tapez une ligne vous pouvez utiliser la touche DEL pour la supprimer. Si vous êtes complètement perdu vous pouvez toutefois presser la touche **CTRL X** pour annuler toute la ligne et recommencer.

MODIFICATIONS

Listez le programme de nouveau (commande **LIST**). L'écran devrait maintenant indiquer :

```
LIST
5 PRINT 3
10 PRINT 1
```

Supposez que nous voulions changer la ligne 10 en :

```
10 PRINT 4
```

Il y a deux méthodes pour faire cela :

1. Entrer de nouveau toute la ligne.
2. Modifier (éditer) la ligne.

Pour une ligne aussi courte et simple que la ligne 10, la première méthode est la plus facile. Tapez :

```
10 PRINT 4
```

et tapez sur RETURN.

Vous voyez que la ligne 10 précédente est maintenant remplacée par la nouvelle ligne 10.

Supposez que vous vouliez changer la ligne 5 par :

5 PRINT 3+6

Cette fois-ci nous utiliserons les facilités de EDIT. Pour éditer une ligne nous devons mettre le curseur au début de cette ligne. Vous pouvez utiliser les touches de contrôle du curseur pour déplacer le curseur jusqu'à la ligne 5. Toutefois il est relativement trop facile de se perdre parmi toutes les lignes imprimées au terminal si vous essayez d'éditer une ligne à la place où elle se trouve. Il est donc plus prudent au moins jusqu'à ce que vous ayez plus de pratique avec la commande EDIT, de déplacer la ligne que vous désirez un espace en-dessous des autres lignes imprimées. Pour ce faire tapez :

EDIT 5

suivi par RETURN.

Appuyez sur **CTRL A** et voyez comment le curseur se déplace le long de la ligne. Souvenez-vous que j'ai mentionné plus hautque les entrées au clavier vont dans un espace de stockage appelé le tampon du clavier qui contient tous ces caractères jusqu'à ce que la touche RETURN ait été enfoncée. Eh bien, le déplacement du curseur sur un caractère à l'écran met ce caractère dans le tampon du clavier tout comme si il avait été entré via le clavier.

Ainsi si vous utilisez **CTRL A** pour déplacer le curseur tout au long de la ligne 5 jusqu'à ce qu'il soit sur l'espace à la droite du dernier caractère comme ci-dessous :

5 PRINT 3 ■

alors le tampon du clavier contiendra "5 PRINT 3" tout comme si vous l'aviez tapé.

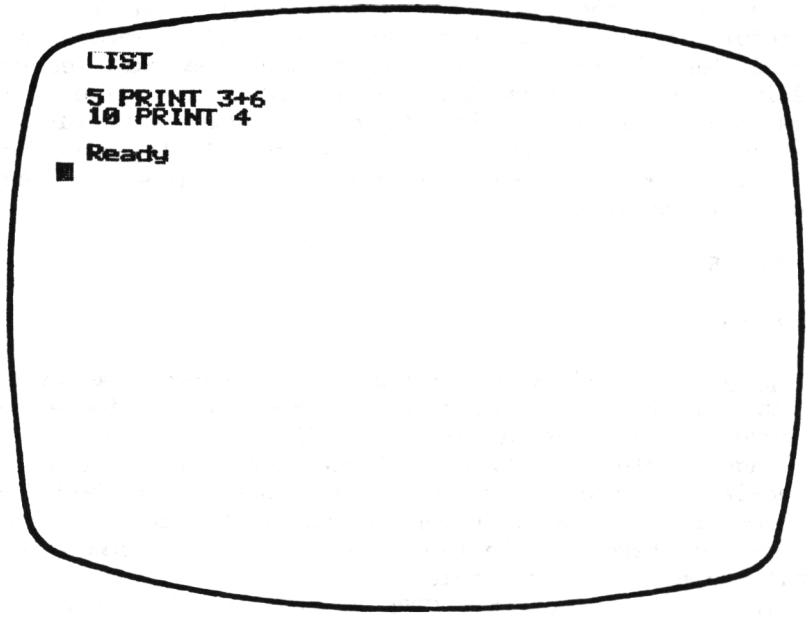
Tapez +6. Ceci apparaîtra à la fin de la ligne de telle manière qu'elle se lise :

5 PRINT 3+6 ■

Les caractères supplémentaires entrés sont aussi allés dans le tampon du clavier. Ce tampon contient maintenant tous les caractères requis, donc appuyez sur RETURN.

Pour vérifier que cela a fonctionné videz l'écran et listez le programme. Celui-ci doit se lire :

```
5 PRINT 3+6
10 PRINT 4
```



Supposez que vous vouliez éditer une ligne de telle manière qu'une partie en soit supprimée. Vous pouvez faire cela par l'usage précautionneux de CTRL A et des touches de contrôle du curseur.

En déplaçant le curseur sur un caractère à l'écran avec CTRL A, cela le copie dans le tampondu clavier. Déplacer le curseur par dessus un caractère avec la touche de contrôle du curseur ne copie pas ce caractère dans le tampon du clavier.

Par exemple, supposez que vous vouliez changer la ligne 5 en :

```
5 PRINT 6
```

c'est-à-dire que vous vouliez enlever les caractères "3+".

Déplacez le curseur à gauche de la ligne. Si vous le souhaitez vous pouvez utiliser EDIT 5 comme précédemment, ou simplement utiliser la ligne 5 déjà sur écran.

Déplacez le curseur le long de la ligne en utilisant CTRL A jusqu'à ce qu'il soit au-dessus du caractère "3". Cela signifie que le tampon du clavier contient les caractères :

```
5 PRINT
```

Utilisez la touche de contrôle du curseur vers la droite pour déplacer le curseur jusqu'à ce qu'il soit au-dessus du caractère "6". Parce que déplacer le curseur avec le caractère de contrôle du curseur ne copie pas le caractère dans le tampon du clavier, le contenu du tampon du clavier n'en sera pas altéré.

Finalement utilisez CTRL A pour déplacer le curseur sur le caractère "6". Cela copie ce caractère dans le tampon du clavier qui contient :

```
5 PRINT 6
```

C'est ce que nous voulions, donc appuyez sur RETURN.

Nettoyez l'écran et listez votre programme. On doit lire :

```
5 PRINT 6
10 PRINT 4
```

Attention - Quand vous changez quelque chose dans le milieu d'une ligne, il est très facile d'oublier de copier le reste de la ligne à l'aide de CTRL A. S'il vous plaît faites attention à cette erreur.

Tenez-vous bien, nous arrivons à des parties plus difficiles :

Supposez que vous vouliez insérer quelques caractères dans le milieu d'une ligne sans rien enlever de ce qui s'y trouve déjà. A première vue ce n'est pas facile. Quand vous tapez de nouveaux caractères ils remplacent les autres caractères de la ligne. A nouveau le secret est dans l'utilisation des touches de contrôle du curseur ainsi que de CTRL A.

Ceci sera mieux expliqué par un exemple. Supposez que nous voulions changer la ligne 10 en :

```
10 PRINT 1+3+4
```

Quand vous insérez des caractères dans une ligne vous avez besoin d'un peu d'écran vide pour travailler dessus, donc utilisez EDIT.

Tapez EDIT 10 suivi par RETURN.

La ligne 10 doit se lire :

10 PRINT 4

Déplacez le curseur sur la ligne avec CTRL A jusqu'à ce qu'il soit sur le caractère "4".
Déplacez le curseur une ligne plus bas avec la touche de contrôle du curseur (flèche vers le bas) et tapez :

1+3+

Parce que le déplacement du curseur sur les caractères à l'écran à l'aide de CTRL A et le fait de les taper ont le même effet, c'est-à-dire de les mettre dans le tampon du clavier -le tampon du clavier contient :

10 PRINT 1+3+

Déplacez le curseur vers le haut et sur la gauche en utilisant les touches de contrôle du curseur jusqu'à ce qu'il soit à nouveau sur le caractère "4". Utilisez CTRL A pour "copier" ce caractère dans le tampon du clavier qui contient maintenant :

10 PRINT 1+3+4

```
LIST

5 PRINT 6
10 PRINT 4

Ready
EDIT 10

10 PRINT 4
           1+3+
```

C'est ce que nous voulions, donc appuyez sur RETURN.

Nettoyez l'écran et LISTEZ votre programme. On doit maintenant lire :

```
5 PRINT 6
10 PRINT 1+3+4
```

Faites exécuter le programme (tapez RUN et appuyez sur RETURN) pour obtenir le résultat :

```
6
8
```

L'utilisation de CTRL A est une puissante méthode de MODIFICATION. Cela veut dire que vous pouvez faire une ligne de programme en copiant n'importe quoi sur l'écran. Il est toutefois assez difficile de s'habituer à cette méthode spécialement parce que vous ne pouvez pas voir ce qui est dans le tampon du clavier pendant que vous êtes en train de MODIFIER la ligne.

Ne vous tracassez pas si vous êtes perdu au départ - un peu de pratique et vous aurez bientôt raison de cela. Vérifiez le résultat de votre ÉDITION en vidant l'écran et en LISTANT le programme.

Attention - Si vous êtes complètement perdu pendant que vous ÉDITEZ une ligne, souvenez-vous d'utiliser CTRL X pour annuler ce que vous avez tapé. Si vous appuyez sur RETURN vous obtiendrez probablement une ligne "complètement absurde" et vous serez bien plus mal en point que lorsque vous avez commencé.

Si vous changez le numéro de ligne pendant que vous ÉDITEZ une ligne, vous créerez une nouvelle ligne et vous n'altérerez pas la ligne avec laquelle vous avez commencé. C'est une méthode très utile pour créer de nouvelles lignes dont le contenu ne diffère pas beaucoup de lignes déjà existantes.

A nouveau cela se comprend mieux à l'aide d'un exemple. Supposez que nous voulions ajouter à notre programme existant la ligne :

```
20 PRINT 1+3+6
```

Utilisez les touches de contrôle du curseur pour déplacer celui-ci au début de la ligne 10 et tapez le caractère "2". Cela s'écrira à la place du "1" et le curseur sera maintenant sur le caractère "0".

Déplacez le curseur le long de la ligne en utilisant CTRL A jusqu'à ce qu'il soit sur le caractère final "4", tapez un "6".

Le tampon du clavier contient maintenant les caractères :

```
20 PRINT 1+3+6
```

C'est ce que nous voulions, donc appuyez sur RETURN.

Notez que la ligne 10 n'est pas changée. Nous avons utilisé son image sur l'écran pour créer la ligne 20 mais cela ne change pas la ligne 10 du programme en mémoire.

La ligne 10 aurait été changée si vous aviez gardé le même numéro de ligne pendant que nous éditions la ligne.

Nettoyez l'écran et listez le programme, il devrait se lire :

```
5 PRINT 6
10 PRINT 1+3+4
20 PRINT 1+3+6
```

Faites exécuter le programme pour obtenir le résultat :

```
6
8
10
```

NOTE - quand vous éditez une ligne de programme après avoir utilisé la commande EDIT vous pouvez trouver que le curseur n'est pas au début de la ligne. Cela arrive quand la ligne prend plus d'une rangée sur l'écran. Spécialement avec la machine V1.0. Utilisez les touches de contrôle du curseur pour positionner le curseur avant l'édition.

NUMEROTATION DES LIGNES

Vous vous souviendrez que la première ligne que vous avez rentrée avait comme numéro de ligne 10 plutôt que 1. Vous vous êtes peut-être rendu compte de la raison pour laquelle cela avait été fait. Cela vous a permis d'insérer une ligne avant la ligne 10. Habituellement nous numérotons les lignes de 10 en 10 (10,20,30 ...), si bien que si des lignes supplémentaires sont nécessaires elles peuvent être insérées facilement.

SUPPRESSION D'UNE LIGNE

Si vous souhaitez supprimer une ligne dans un programme, tapez son numéro de ligne suivi par RETURN.

Par exemple, tapez 5 et appuyez sur RETURN.

Videz l'écran et LISTEZ le programme.

Vérifiez que la ligne 5 a bien disparu.

Comme deuxième contrôle faites exécuter le programme pour obtenir les résultats :

```
8
10
```

NEW

Nous en avons fini avec ce programme. Nous allons voir comment s'en débarrasser. Nous pourrions supprimer chaque ligne une par une mais il y a un moyen plus facile.

Tapez NEW et appuyez sur RETURN.

Effacez l'écran et listez (LIST). Le programme est parti, rien n'est LISTE.

La commande NEW est utilisée pour enlever tout programme en mémoire et donner au programmeur "une ardoise nette".

Faites attention quand vous utilisez cette commande. Soyez bien sûr que vous voulez réellement enlever tout votre programme de la mémoire.

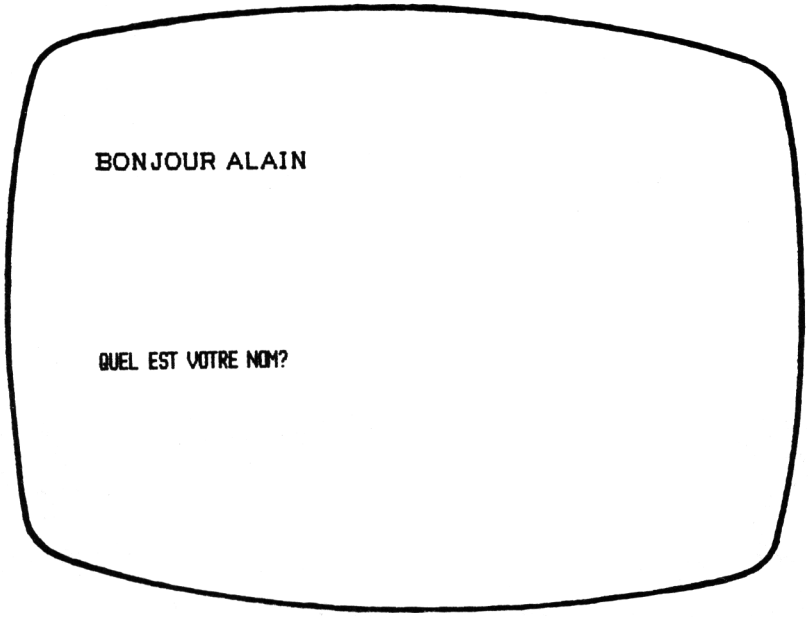
UNE RECREATION

Vous avez travaillé dur avec EDIT et j'espère que vous ne vous êtes pas trop ennuyé. Avant de partir, nous allons nous amuser un peu. Tapez ce programme, souvenez-vous d'appuyer sur RETURN après chaque ligne, n'essayez pas de le comprendre à ce niveau.

```
10 PAPER 0:INK 3:CLS
20 REPEAT
30 PRINT:PRINT:PRINT:PRINT
40 INPUT "QUEL EST VOTRE NOM";A$
50 IF A$="ANNE" THEN A$="ANNE CHERIE"
60 PRINT CHR$(12)
70 PRINT CHR$(4):SPC(10)CHR$(27):"JBONJOUR";A$
80 PRINT CHR$(4)
90 UNTIL 0.
```

La ligne 70 prendra deux lignes sur l'écran mais cela n'a pas d'importance.

Faites exécuter le programme. Tapez votre nom quand cela vous est demandé, suivi par RETURN.

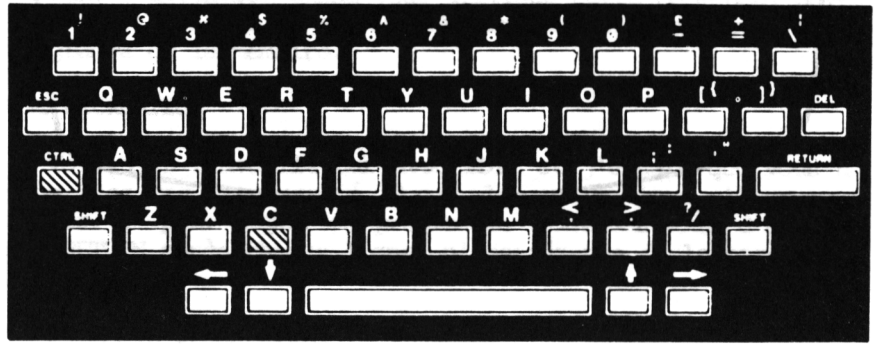


Le résultat obtenu en tapant "ANNE" (suivi par RETURN) est intéressant. Ceci est, comme vous l'avez deviné, le prénom de ma femme. Mais je suis tranquille car elle ne lit jamais mes livres ! Essayez de changer la ligne 50 pour répondre à vos propres besoins.

Je m'excuse auprès des lectrices qui s'appellent ANNE, honnêtement vous n'êtes pas concernées !

ARRET

Une fois que vous êtes fatigué du programme vous voudrez l'arrêter. Vous vous souviendrez que j'ai mentionné CTRL C dans le chapitre précédent et que j'ai dit que vous l'utiliseriez à travers tout ce livre. C'est le premier exemple. Arrêtez le programme en utilisant CTRL C quand l'ordinateur vous demande votre nom. Ne supprimez pas le programme maintenant, car nous allons l'utiliser dans le prochain chapitre.



CTRL C - Pour sortir du programme

Chapitre 4

CHARGEMENT ET SAUVEGARDE DES PROGRAMMES

J'ai mis ce chapitre près du début de ce livre de manière tout à fait délibérée. Pourquoi, vous demandez-vous probablement, apprendre à sauvegarder des programmes sur bande avant même de savoir les écrire ? Pour cela il y a deux raisons :

1. Taper de petits programmes tout le temps est fastidieux et conduit à des erreurs. Vous devriez prendre l'habitude de sauvegarder tout ce que vous voulez utiliser plus tard aussitôt que vous l'avez mis au point (qu'ils fonctionnent correctement). Ainsi vous n'aurez pas à les mettre au point plusieurs fois de suite. Il est utile de sauvegarder même des programmes incorrects si vous êtes interrompu lors de leur mise au point.

2. Il y a un certain nombre de bandes contenant d'excellents programmes commerciaux disponibles pour l'ORIC. Vous écrirez bientôt vos propres programmes mais cela ne vous empêchera pas d'acheter, d'utiliser et de vous amuser à l'aide de ces logiciels écrits de manière professionnelle.

ENREGISTREUR DE CASSETTES

Vous aurez besoin d'un enregistreur de cassettes. Vous pouvez décider d'en utiliser un que vous possédez déjà ou bien en acquérir un spécialement pour l'utiliser avec l'ORIC.

Vous n'avez pas besoin d'un enregistreur de cassettes très onéreux avec stéréo et contrôle de tonalité. Si vous disposez de ces possibilités alors réglez l'équilibre stéréo de manière à faire l'enregistrement sur un seul canal et réglez le contrôle de tonalité pour donner le maximum d'aigus et le minimum de graves.

Vous trouverez qu'un compteur d'enregistrement est extrêmement utile. En fait j'irai jusqu'à dire que si votre machine actuelle n'a pas cette possibilité vous devriez considérer sérieusement l'achat d'une machine qui possède ce dispositif.

Le connecteur de cassettes à l'arrière de l'ORIC consiste en une prise ronde à sept trous connue sous le nom de prise DIN. Le connecteur sur votre enregistreur de cassettes peut être l'un des suivants :

1. Une prise DIN à trois trous.

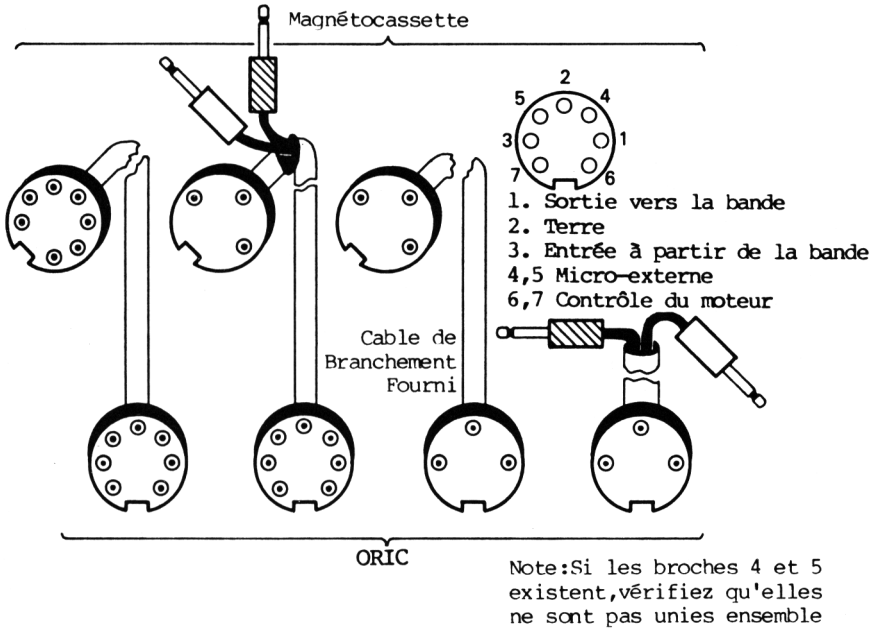
2. Une prise DIN à cinq trous.

3. Deux prises mâles marquées EAR et MIC.

4. L'une quelconque des précédentes plus une prise de contrôle de moteur.

Si votre enregistreur a une prise DIN à trois ou cinq trous, alors utilisez le connecteur DIN trois aiguilles, fourni avec votre machine. **N'utilisez pas** un connecteur à cinq trous.

Si vous avez un tout autre connecteur sur votre enregistreur, alors prenez conseil auprès de votre distributeur d'ordinateurs. Le contrôle du moteur (arrêt et redémarrage automatique) est un dispositif utile et l'ORIC fournit le moyen de l'actionner. Faites attention, certains enregistreurs de cassettes ont des moteurs trop puissants pour que l'ORIC puisse les contrôler. Là aussi, votre distributeur d'ordinateurs devrait être capable de vous conseiller en ce domaine.



Connecteurs DIN

CASSETTES

Les programmes ORIC sont sauvegardés sur des cassettes audio ordinaires - le même genre de cassettes que vous utiliseriez pour enregistrer de la musique. Je paye toujours un peu plus pour obtenir des cassettes de bonne qualité avec un faible niveau de parasites qui selon moi donnent un accroissement intéressant de la fiabilité. Vous pouvez préférer utiliser des bandes très courtes (C10 ou C15) proposées par les distributeurs d'ordinateurs. Celles-ci peuvent vous permettre de gagner du temps quand il s'agit de chercher les programmes sur bandes.

TERMES TECHNIQUES

Quand je dis **sauvegarder** un programme je veux dire enregistrer le programme, qui est dans la mémoire de l'ordinateur, sur une cassette.

Quand je parle de **charger** un programme, je veux dire lire le programme depuis la cassette et le copier dans la mémoire de l'ordinateur. C'est-à-dire remettre dans la mémoire de l'ordinateur un programme qui a été sauvegardé sur cassette quelque temps auparavant.

CSAVE

Pour sauvegarder un programme sur cassette, ORIC utilise la commande CSAVE (ce qui est une abréviation pour Cassette Save).

Nous allons sauvegarder sur bande le programme que nous avons tapé dans le chapitre précédent. Si vous avez coupé l'alimentation de l'ordinateur entre les chapitres, alors j'ai bien peur que vous ayez à retaper le programme.

Mettez une nouvelle cassette dans l'enregistreur et rembobinez la au début. Mettez le compteur de bande à 0.

Utilisez le défilement "avant accéléré" pour amener la bande jusqu'à ce que le compteur soit à peu près à 005. Cela permet de s'assurer que vous n'allez pas enregistrer sur l'amorce de bande (le morceau de plastique au début de la bande).

Assurez-vous que le cordon entre l'ordinateur et l'enregistreur de cassettes n'est pas près du cordon d'alimentation ou du cordon relié à la télévision. Autrement cela pourrait créer des interférences.

Pour sauvegarder un programme vous devez lui donner un nom. Ce nom peut être tout ce que vous souhaitez à condition qu'il n'ait pas plus de 17 caractères (machine V1.0) ou 16 caractères (machine V1.1). Appelons notre premier programme DEMO1.

Tapez CSAVE "DEMO1", S

Déclenchez l'enregistrement de votre enregistreur de cassettes (si votre enregisreur a un démarrage automatique vous n'avez pas besoin de le faire) et appuyez sur RETURN.

Après quelques secondes le message "SAVING DEMO1" (ou "SAVING DEMO1 B") apparaîtra en haut de l'écran. Quand la sauvegarde est complète le mot "READY" apparaît sur l'écran. Arrêtez l'enregistrement de la cassette.

CLOAD

Utilisez la commande **NEW** pour effacer le programme contenu dans la mémoire. Nous allons voir comment ramener le programme en mémoire depuis la bande. Pour ce faire l'ORIC utilise la commande CLOAD (abrégé pour Cassette Load).

Rembobinez la bande jusqu'à ce que le compteur indique un nombre un peu inférieur à celui auquel vous avez commencé à sauvegarder le programme (000 fera très bien l'affaire dans ce cas). Tapez CLOAD"DEMO1",S en prenant soin à ce que le nom DEMO1 soit exactement le même nom utilisé quand vous avez sauvegardé, car les espaces ont une importance ici.

Appuyez sur RETURN et commencez la lecture de l'enregistrement. Le message "SEARCHING.." devrait apparaître en haut de l'écran, suivi par le message "LOADING DEMO1" (ou "LOADING DEMO1 B") quand le début du programme est détecté. Quand le chargement est complet le mot "READY" apparaît sur l'écran. Eteignez l'enregistreur de cassettes et faites exécuter le programme.

PROBLEMES RELATIFS AUX ENREGISTREURS DE CASSETTES

Si le mot "READY" n'apparaît pas, essayez de nouveau la commande CLOAD en réglant le volume différemment sur le magnétophone cassette en vous souvenant de rembobiner la bande entre chaque cassette. Vous pouvez avoir besoin de positionner l'interrupteur RESET en dessous de l'ordinateur pour faire sortir la machine du mode de "recherche". Si votre magnéto-cassettes a des prises EAR et MIC de la même taille, essayez de changer de connection, en les intervertissant.

Vous pouvez obtenir le message d'erreur "FILE ERROR - LOAD ABORTED" quand vous essayez de charger. Cela indique au moins que vous avez quelque chose sur votre bande.

Si vous n'avez toujours pas de succès, essayez de sauvegarder un programme plus petit. Tapez

```
10 PRINT1
```

et appuyez sur RETURN, ensuite exécutez toute la procédure de sauvegarde, effacez le programme en mémoire et cherchez à recharger.

Si vous n'avez toujours pas de succès, essayez de faire fonctionner votre enregistreur sans qu'il soit rattaché à l'ordinateur.

S'il n'y a rien sur la bande, il peut y avoir une faute, soit dans votre connecteur, soit dans l'enregistreur. Testez l'enregistreur en enregistrant autre chose qu'un programme d'ordinateur.

Si vous obtenez un son déplaisant, fort et très aigu, vous avez alors un programme (ou quelque chose de très similaire) sur bande. La tête d'enregistrement de votre magnéto-cassettes peut être désalignée. Référez-vous aux instructions du fabricant pour les ajuster.

RAPIDE ET LENT

Supposons toutefois que rien ne se soit mal passé et que vous ayez enregistré votre programme avec succès. Vous vous demandez probablement à quoi sert le "S" dans les commandes CLOAD et CSAVE.

Quand un ordinateur sauvegarde de l'information ou charge une information à partir d'une bande il le fait à une certaine vitesse. Cela n'a rien à voir avec la vitesse à laquelle la bande se déplace qui reste constante ; c'est la vitesse à laquelle l'information est envoyée sur la ligne entre l'ordinateur et l'enregistreur. Ceci est connu sous le nom de débit exprimé en "baud".

L'ORIC a deux taux de débit. Vous avez sauvegardé et chargé le programme à 300 bauds qui est un taux relativement lent. Le "S" dans la commande signifiait "slow" (c'est-à-dire lentement).

L'ORIC peut aussi sauvegarder et charger des informations à 2400 bauds - 8 fois plus rapide. Cela économise votre temps et signifie aussi que vous pouvez avoir 8 fois plus d'informations sur votre bande. L'inconvénient est que, parce que l'information est stockée d'une manière si dense sur bande, le moindre défaut de la bande ou de l'information qui est dessus peut conduire à un "mauvais" programme -c'est-à-dire qu'il ne se chargera pas. En fait le débit de 300 bauds est bien plus sûr que le débit de 2400 bauds. A la vitesse supérieure (2400 bauds) l'alignement des têtes d'enregistrement et le volume d'enregistrement sont bien plus critiques.

Pour sauvegarder un programme à vitesse normale (2400 bauds), amenez votre bande sur une section vierge (disons que le compteur indique le numéro 30). Suivez exactement la même procédure excepté que la commande que l'on va utiliser est :

```
CSAVE"DEMO1"
```

suivi comme d'habitude par RETURN.

Assurez-vous que votre programme a été sauvegardé en le rechargeant en mémoire. De nouveau la procédure est la même que précédemment excepté que la commande est cette fois :

```
CLOAD"DEMO1"
```

Vous trouverez peut-être que vous avez besoin de plusieurs tentatives successives pour obtenir le meilleur ajustement de volume avant que le programme puisse se charger en mémoire.

Un programme doit être chargé à la même vitesse (300 ou 2400 bauds) que celle à laquelle il a été sauvegardé. Sinon il ne se chargera pas.

VERIFICATION DE PROGRAMMES

Il est pratique de pouvoir vérifier qu'un programme a été correctementsauvegardé pendant que ce programme est encore en mémoire. La machine V1.1 fournit ce dispositif de vérification qui permet à l'ORIC de comparer un programme sur bande avec celui en mémoire sans altérer le programme en mémoire.

Une forme spéciale de la commande CLOAD est utilisée pour vérifier. Elle est :

```
CLOAD"...",V
```

ou CLOAD"...",V,S

suivant que le programme que vous vérifiez a été sauvegardé à 2400 ou 300 bauds.

Cela est mieux expliqué par un exemple. Vous devez voir DEMO1 sauvé à vitesse lente au début de votre bande. Vous devez aussi avoir DEMO1 en mémoire de l'ordinateur, sinon chargez-le.

Rembobinez la bande au début. Tapez :

```
CLOAD"DEMO1",V,S
```

appuyez sur RETURN et démarrez la bande.

Vous obtiendrez le message "Searching" comme précédemment, suivi par "Verifying DEMO1 B". Quand le message :

```
0 verify errors detected

Ready
```

apparaîtra sur l'écran, arrêtez la bande. Vous avez vérifié que le programme sur bande est le même que le programme en mémoire.

Si votre vérification échoue (c'est-à-dire que vous obtenez un nombre non nul d'erreurs de vérification) alors ajustez le volume de votre enregistreur de cassettes et recommencez. Si vous ne pouvez toujours pas vérifier votre programme, assurez-vous que votre machine n'est pas de la version V1.0. **Seule** la version V.1.1 permet la vérification.

Si vous avez fait en sorte de sauver DEMO1 à plus grande vitesse (2400 bauds), essayez aussi de vérifier cette sauvegarde. Utilisez :

```
CLOAD"DEMO1",V
```

La machine V1.1 acceptera aussi

CLOAD"",V,S
et CLOAD"",V

Ces programmes vérifient le contenu du programme en mémoire par rapport au premier programme qu'elle rencontre sur bande.

FUSION DE PROGRAMMES

Les gros programmes sont souvent faits de petits programmes fusionnés entre eux. La machine V1.1 vous permet de joindre un programme sur bande à un programme déjà en mémoire.

Note - Cela ne fonctionne pas sur la machine V1.0.

A nouveau cela est mieux illustré par un exemple. Effacez la mémoire de l'ordinateur en tapant NEW et en pressant RETURN. Rembobinez la cassette de telle manière que vous n'écriviez pas en surimpression sur quelque chose que vous avez déjà sauvegardé.

Tapez ce programme :

```
100 PRINT"SANS TROIS"
```

Faites exécuter ce programme et obtenez le message :

```
SANS TROIS
```

sur l'écran.

Sauvegardez ceci sur bande comme DEMO1A. Vérifiez qu'il a été sauvegardé correctement.

Effacez le programme en mémoire et entrez le programme suivant :

```
10 CLS
20 PRINT
30 PRINT
40 PRINT"JAMAIS DEUX"
50 PRINT
```

Faites exécuter le programme. Vous obtiendrez le message :

```
JAMAIS DEUX
```

sur l'écran.

Rembobinez la cassette en arrière juste avant le départ du programme DEMO1A. Tapez :

```
CLOAD"DEMO1A",J
ou CLOAD"DEMO1A",J,S
```

selon que vous avez sauvegardé DEMO1A à la vitesse normale ou lente. Faites fonctionner la bande.

Quand le chargement est fini (Ready apparaît sur l'écran), listez le programme résultant.

Vous obtiendrez :

```
10 CLS
20 PRINT
30 PRINT
40 PRINT"JAMAIS DEUX"
50 PRINT
100 PRINT"SEE HOW THEY RUN"
```

Faites exécuter le programme pour obtenir le message :

```
JAMAIS DEUX

SANS TROIS
```

DEMO1A s'est fusionné avec le programme en mémoire.

ATTENTION - Quand vous fusionnez deux programmes, assurez-vous qu'aucune des lignes du second programme n'a le même numéro de ligne que l'une des lignes du premier programme. Sinon, le programme résultant ne fonctionnera pas correctement.

RECHERCHE DES PROGRAMMES

Quand vous sauvegardez un programme sur bande, écrivez toujours le nom du programme, le numéro du compteur auquel il commence et s'il a été enregistré à vitesse normale ou lente. Cette information peut être gardée avec la cassette, la plupart des cassettes sont vendues avec une carte sur laquelle son contenu peut être noté.

Si vous ne savez pas si un programme est sur une bande, mais que vous connaissiez son nom, alors la commande CLOAD avec le nom spécifié fera que l'ordinateur ignorera tous les autres programmes et chargera seulement le programme de ce nom.

Par exemple, supposez que vous sachiez qu'un programme appelé CACHECACHE est quelque part sur la bande et que ce programme a été enregistré à la vitesse de 300 bauds. Pour trouver ce programme vous pouvez rembobiner la bande au début et taper :

```
CLOAD"CACHECACHE",S
```

puis appuyez sur RETURN. Si vous faites dérouler la bande l'ordinateur ignorera tous les programmes jusqu'à ce qu'il arrive sur celui appelé CACHECACHE qu'il chargera alors en mémoire. La machine V1.1 indiquera en haut de l'écran toute autre chose (les autres programmes) trouvée sur la bande pendant qu'il cherche le programme nommé CACHECACHE.

Si au contraire vous savez où est un programme mais que vous ayez oublié son nom, alors la commande :

```
CLOAD""(ou CLOAD"",S)
```

fera que l'ordinateur chargera en mémoire le premier programme qu'il trouvera sur la bande. Notez qu'il n'y a pas d'espace entre les guillemets.

DEMARRAGE AUTOMATIQUE

Quand vous avez chargé DEMO1 dans l'ordinateur, vous avez eu à taper RUN suivi par RETURN pour démarrer le programme. Cela pourrait être pratique que le programme démarre aussitôt qu'il est chargé. Pour cela, nous ajoutons "AUTO" à la commande CSAVE quand nous sauvegardons le programme.

Si DEMO1 n'est pas dans la mémoire de l'ordinateur alors chargez-le dedans.

Sauvegardez-le de nouveau, cette fois en utilisant la commande:

CSAVE"DEMO1",S,AUTO
si vous sauvegardez à la vitesse de 300 bauds et

CSAVE"DEMO1",AUTO
si vous sauvegardez à la vitesse de 2400 bauds.

Rechargez le programme de manière normale. Il devra maintenant s'exécuter aussitôt qu'il aura été chargé.

SAUVEGARDE DES BLOCS DE MEMOIRE

Nous avons discuté de la sauvegarde des programmes contenus dans ce que nous avons appelé la "mémoire des programmes". L'ORIC a différentes zones de mémoires qui sont utilisées pour stocker différents types d'informations. Par exemple, il y a des parties de mémoire qui contrôlent ce que l'ordinateur met sur l'écran, tandis que d'autres parties peuvent être utilisées pour des programmes ou des données.

Il est possible de sauvegarder sur bande l'information contenue dans ces sections spécifiques de la mémoire et de recharger cette information dans ces sections ultérieurement. Nous étudierons cela plus en détail au chapitre 14.

SAUVEGARDE DE LONGS PROGRAMMES

Il y a peu de choses plus ennuyeuses que de taper un long programme -disons tiré d'un magazine- et de le perdre du fait d'un défaut d'alimentation ou parce que l'ordinateur ne veut pas le sauvegarder. ORIC ne m'a pas fait cela jusqu'à maintenant, mais la plupart des autres ordinateurs le font ! La manière d'éviter l'horrible sensation qui consiste à voir deux heures de travail s'évanouir, partir en fumée, est de sauvegarder le programme toutes les vingt lignes, ainsi, lorsque l'alimentation électrique a un hoquet à la ligne 2000, eh bien les lignes 10 à 1800 sont sauvegardées sur bande.

A moins que vous n'ayez des actions dans la compagnie qui les fabrique, vous n'avez pas besoin d'utiliser des bandes vierges chaque fois que vous sauvegardez le même programme. Normalement, vous rembobinez la bande à chaque fois et utilisez la même section de bande pour chaque sauvegarde.

Si vous êtes du type prudent, vous aurez probablement la vision d'un à-coup d'alimentation électrique survenant au milieu de la sauvegarde des lignes 10 à 2000, si bien que non seulement le programme est perdu mais la version précédente des lignes 10 à 1800 est perdue aussi. Pour l'éviter, car cela se peut et cela arrive, nous utilisons le principe du "back-up" (copie de secours).

Cela suppose que l'on sauve les lignes 10 à 200 sur une partie de la bande, les lignes 10 à 400 sur une autre, les lignes 10 à 600 sur la première section (écrivant ainsi par-dessus les lignes 10 à 200), les lignes 10 à 800 sur la seconde section (écrivant ainsi par-dessus les lignes 10 à 400), et ainsi de suite. De cette manière, même si un désastre survient au moment de l'opération de sauvegarde, vous ne perdrez jamais plus de 20 lignes de code.

ENCORE QUELQUES RECREATIONS

Vous savez comment utiliser le clavier et comment sauvegarder et charger des programmes. Le meilleur moyen de s'assurer que vous vous souvenez de ce que vous avez appris, est de faire un peu de pratique. Voici deux programmes que vous devez taper, sauvegarder et exécuter.

Comme précédemment, je vous demande de me croire sur parole, et d'utiliser ces programmes comme amusement.

DEMO2

Ce programme démontre les capacités de l'ORIC du point de vue couleur. Je vous suggère que vous le nommiez DEMO2 quand vous le sauvegarderez.

Faites exécuter le programme en tapant la commande RUN suivi par RETURN, arrêtez par CTRL C. Ce programme cache le curseur pour améliorer l'affichage de l'écran. Vous pouvez le ré-obtenir avec CTRL Q, après que vous ayez arrêté le programme à l'aide de CTRL C.

```
10 CLS:PAPER1
20 FOR N=0 TO 7
30 PLOT8,5+N,23-N
40 PLOT28,5+N,17
50 READ A$
60 PLOT10,5+N,A$
70 PLOT9,5+N,N
80 NEXT
90 PRINT CHR$(17)
100 GOTO 100
110 DATA NOIR SUR BLANC
120 DATA ROUGE SUR CYAN
130 DATA VERT SUR MAGENTA
140 DATA JAUNE SUR BLEU
150 DATA BLEU SUR JAUNE
160 DATA MAGENTA SUR VERT
170 DATA CYAN SUR ROUGE
180 DATA BLANC SUR NOIR
```

DEMO3

Je suggère que vous nommiez le programme suivant DEMO3. Je trouve ce programme très utile car j'ai un petit garçon de 6 ans qui l'utilise pour ses devoirs. Arrêtez le programme à l'aide de CTRL C, revenez à l'impression simple hauteur à l'aide de CTRL D, et remettez le curseur à l'aide de CTRL Q.

Quand vous taperez le programme, vous trouverez que les lignes 80, 100, 110 et 120 occupent deux lignes sur l'écran. Cela n'affecte pas le programme.

```
10 PRINT CHR$(17)
20 X%=RND(1)*11+10
30 Y%=RND(1)*9+1
40 PAPER0:INK3
50 PRINT CHR$(12)
60 FOR N=0 TO 7
70 PRINT:NEXT
80 PRINT CHR$(4);CHR$(27);"J";SPC(13);X%-Y%;"+";Y%;"=";
90 INPUT A$
100 IF VAL(A$)=X% THEN 110 ELSE EXPLODE:PRINT CHR$(4):
     GOTO 50
110 PRINT:PRINT:PRINT:PRINT CHR$(27);"J";SPC(18);"EXACT"
120 FOR N=0 TO 9:ZAP:WAIT 5:NEXT:PRINT CHR$(4):RUN 20
```

Chapitre 5

EXPLORATION DE L'ORIC

CARACTERISTIQUES GENERALES

Dans ce chapitre je vais décrire quelques-unes des caractéristiques de fonctionnement de l'ORIC et quelques instructions très utiles que vous verrez dans beaucoup de programmes d'ordinateur. A cette étape je donnerai des explications brèves et je donnerai plus de détails sur certaines de ces caractéristiques et instructions à des étapes ultérieures plus appropriées.

Tapez NEW et appuyez sur RETURN pour vider la mémoire contenant le programme. Si vous le souhaitez vous pouvez laisser l'ordinateur avec ses coloris d'avant-plan et d'arrière-plan actuels. Si vous préférez travailler à l'encre noire sur fond blanc, tapez INK0:PAPER 7 et appuyez sur RETURN.

REM

L'instruction REM fait que l'ordinateur ignore toutes choses (même des instructions) qui la suivent jusqu'à ce que la fin de la ligne soit atteinte.

Vous pouvez vous demander quel est l'intérêt d'une instruction que l'ordinateur ignore. La réponse est que ce ne sont pas seulement les ordinateurs qui lisent les programmes, les gens aussi les lisent ! Cela ne vous étonnera pas trop d'apprendre que tous les programmes ne marchent pas du premier coup. Le programmeur doit corriger les erreurs pour que le programme puisse marcher. Et probablement qu'il y reviendra à une date ultérieure pour l'améliorer. S'il vous plaît croyez-moi (mes propos sont fondés sur une longue et amère expérience), c'est beaucoup plus facile si des instructions REM sont utilisées pour indiquer ce que chaque partie du programme fait.

Ceci est un exemple un peu excessif. Vous n'avez pas besoin de le taper à l'ordinateur à moins que vous ne soyez vraiment très zélé.

```
10 REM CECI ILLUSTRE LES POSSIBILITES DE L'INSTRUCTION
   REM. JE PEUX TAPER DES INSTRUCTIONS TELLES QUE ZAP,
   PING, EXPLODE, PRINT 1, ETC. APRES REM MAIS
   L'ORDINATEUR LES IGNORE. SI JE VEUX UNE EXPLOSION JE
   DOIS ALLER A LA LIGNE 20 ET TAPER
20 EXPLODE
```

L'instruction REM peut être mise au milieu d'une ligne après une instruction à condition qu'elle soit précédée par deux points (:).

Par exemple :

```
10 PING:REM CECI FAIT UN BRUIT
```

Si vous utilisez l'instruction REM dans le milieu d'une ligne, vous pouvez remplacer tous les :REM par une apostrophe (').

Par exemple :

```
10 PING'CECI FAIT UN BRUIT
```

Vous pouvez utiliser cette abréviation seulement dans le milieu d'une ligne après une instruction. Vous ne pouvez l'utiliser au début. Par exemple :

```
10 'CECI PRODUIT UNE ERREUR
```

fait justement cela !

Faites attention, n'oubliez pas que tout ce qui suit REM est ignoré jusqu'à ce que vous atteigniez la fin de la ligne.

Par exemple :

```
10 PING:REM UN BRUIT:ZAP:REM UN AUTRE BRUIT
```

ne fera que "ping". Cela ne fera pas "zap" car tout ce qui est après le premier REM est ignoré.

DEUX POINTS (:)

Nous avons vu l'utilisation de DEUX POINTS dans le dernier exemple. Cela vous permet de mettre plus d'une instruction sur la même ligne. C'est particulièrement utile dans les instructions IF...THEN que nous étudierons au chapitre 8.

STOP

Effacez l'espace mémoire avec NEW et effacez l'écran. Tapez ce programme :

```
10 REM UN PROGRAMME DE DEMONSTRATION
20 PRINT 1
30 PRINT 2
40 PRINT 3
50 PRINT 4
```

Faites exécuter ce programme. Une fois qu'il est complet, tapez :

```
35 STOP
```

et appuyez sur la touche RETURN. Effacez l'écran et faites tourner le programme de nouveau. Notez que cette fois-ci seulement les deux premiers nombres sont imprimés. La commande STOP arrête le programme.

CONT

Tapez CONT, et appuyez sur RETURN.
Le programme continuera de la ligne 40 jusqu'à la fin.
CONT est utilisé pour redémarrer un programme après qu'il ait rencontré une commande STOP. Le programme repartira à la première instruction après la commande STOP. CONT peut être utilisé seulement comme une commande immédiate. Ne l'utilisez pas comme instruction dans un programme.

GOTO

GOTO dans un programme, suivi par un numéro de ligne fait que l'ordinateur va à cette ligne et exécute ce qui est indiqué à cet endroit.
Ajoutez cette ligne au programme précédent (n'oubliez pas d'appuyer sur RETURN après que vous ayez tapé la ligne).

```
60 GOTO 20
```

Effacez l'écran et faites exécuter le programme. Comme auparavant il imprimera les deux premiers nombres et S'ARRÊTERA. Tapez CONT (autrement dit tapez CONT et appuyez sur RETURN). Le programme imprimera les deux nombres suivants. Ensuite il retournera au début et imprimera de nouveau les deux premiers nombres.
L'effet de l'instruction GOTO est de faire repartir le programme au début, après que la ligne 50 ait été exécutée, de telle manière que les lignes 20 et 30 soient exécutées de nouveau.
GOTO peut être utilisée comme une commande immédiate plutôt que comme une instruction. Ceci peut être utile si vous souhaitez commencer l'exécution d'un programme à un endroit quelconque de celui-ci. Dans le programme de démonstration donné, essayez les effets de GOTO 30, GOTO 40 ou GOTO 50 au lieu de RUN.

NOTE - Vous pouvez aussi utiliser RUN 30, RUN 40 ou RUN 50 au lieu de GOTO 30 etc...

BOUCLE INFINIE

Supprimez la ligne 35 en tapant 35 et en appuyant sur RETURN.

Effacez l'écran et LISTEZ le programme. Vous obtiendrez :

```
10 REM UN PROGRAMME DE DEMONSTRATION
20 PRINT 1
30 PRINT 2
40 PRINT 3
50 PRINT 4
60 GOTO 20
```

Etudiez ce programme attentivement. Vous verrez qu'à chaque fois qu'il arrive à la ligne 60, il fait une boucle en arrière vers la ligne 20. Ainsi le programme exécutera les instructions des lignes 20 à 60 continuellement. Cela est dénommé une boucle infinie.

Faites exécuter ce programme pour voir l'effet produit par une boucle infinie. Arrêtez le programme par CTRL C.

DEFILEMENT

La boucle infinie fait que l'affichage à l'écran "défile". Cela signifie que, quand la ligne du bas de l'écran est atteinte, la prochaine impression sur l'écran fera que toutes les lignes précédemment imprimées se déplaceront d'un cran vers le haut et les impressions sur la ligne du haut disparaîtront de l'écran.
Un autre exemple de défilement arrive quand nous listons un programme qui est trop long pour tenir sur un seul écran.

Effacez la mémoire programme avec NEW et effacez l'écran. Tapez ce programme :

```
10 REM CECI EST UN PROGRAMME STUPIDE
20 REM QUI NE FAIT RIEN
30 REM
40 REM
50 REM
60 REM
70 REM
80 REM
90 REM
100 REM
110 REM
120 REM
130 REM
140 REM
150 REM
160 REM
170 REM
180 REM
190 REM
200 REM
210 REM
220 REM
230 REM
240 REM
250 REM
260 REM
270 REM
280 REM
290 REM
300 REM
```

Vous verrez que quand vous entrez les dernières lignes de ce programme les lignes du début sont "perdues" et disparaissent du haut de l'écran. Ne vous inquiétez pas la machine les a toujours en mémoire. La raison pour laquelle les lignes du haut sont perdues de vue est que la machine ne peut afficher que 26 lignes d'impression sur l'écran à un moment donné.

Quand vous LISTEZ un programme il défilera toujours vers le haut de l'écran et les lignes du haut seront toujours perdues. En fait dans ce cas la situation est même pire. Car l'ordinateur imprime une ligne blanche et ensuite le mot "READY" sur les deux lignes du bas de l'écran, deux lignes supplémentaires sont donc perdues dans le haut.

Pour arrêter le défilement de manière à ce que vous puissiez regarder le listing du programme, appuyez sur la barre d'espacement. Pour redémarrer le défilement du listing appuyez sur n'importe quelle touche (sauf SHIFT ou CTRL).

Si vous le désirez vous pouvez lister une seule ligne sur l'écran. Par exemple, effacez l'écran et tapez :

```
LIST 50
```

La ligne 50 sera imprimée sur l'écran.

Probablement plus utile est la possibilité de lister une partie de programme, disons des lignes 20 à 60. Pour faire ceci, effacez l'écran et tapez :

```
LIST 20-60
```

CLS

Nous savons comment effacer l'écran par CTRL L. Il y a aussi une instruction qui fait la même chose. Tapez CLS et appuyez sur RETURN. CLS signifie "Clear Screen" (effacer l'écran). A la différence de CTRL L, CLS peut être utilisé comme instruction dans les programmes.

FOR...NEXT

La boucle **FOR...NEXT**, qui sera étudiée plus en détail au chapitre 8, permet de répéter un même processus un certain nombre de fois, mais avec des valeurs différentes à chaque fois. C'est un moyen pratique de constater l'effet produit par différentes valeurs sans avoir à réécrire le programme à chaque fois. Les deux lignes

```
10 FOR N=0 TO 3

100 NEXT N
```

répéterons quatre fois les instructions comprises entre les 10 et 100 en donnant successivement à N les valeurs 0, 1, 2 et 3 à chaque endroit du programme où N est mentionné. "NEXT N" à la ligne 100 peut être abrégé en "NEXT".

LET

L'instruction **LET** est utilisée pour qu'une chose soit égale à une autre. Quand vous écrirez des programmes, vous utiliserez parfois ce qui est appelé des **"variables"**. Comme le nom l'indique, ce sont des quantités qui varient durant l'exécution du programme. Ces variables sont habituellement représentées par des lettres.

Au commencement de l'exécution du programme, une variable doit être "initialisée", c'est-à-dire qu'elle doit recevoir le nombre par lequel elle commence (si ce n'est pas un zéro). C'est l'une des utilisations de l'instruction **LET**.

Par exemple :

```
LET A=1
LET HAUTEUR=6
```

LET peut aussi être utilisé pour établir un compteur ou pour additionner ensemble deux variables. Par exemple :

```
LET B=B+1
LET C=N+M
```

Nous étudierons les variables plus en détail aux chapitres 8 et 9; ne vous inquiétez donc pas si vous êtes un peu perplexe.

"LET" peut être omis - et l'est souvent - dans l'instruction d'affectation, si bien qu'au lieu de "LET A=6", nous pouvons taper "A=6".

WAIT

Une autre instruction utile est WAIT suivi par un nombre. Cela arrête toute action pour un temps "proportionnel à" ce nombre, WAIT 100 donnant à peu près un délai d'une seconde, WAIT 50 donnant un délai d'à peu près une demi-seconde, et ainsi de suite.

INK et PAPER

Vous avez utilisé ces instructions précédemment et j'espère qu'elles sont facilement compréhensibles. INK, suivie par un nombre entier compris entre 0 et 7 inclusivement, établit la couleur du texte sur l'écran, c'est-à-dire "celle de l'encre", ou de l'avant-plan. PAPER suivi par un nombre entier compris entre 0 et 7 inclusivement établit la couleur du "papier" ou de l'arrière-plan.

Les nombres correspondants à chaque couleur à la fois pour INK et PAPER sont :

NOMBRE	COULEUR
0	NOIR
1	ROUGE
2	VERT
3	JAUNE
4	BLEU
5	MAGENTA (POURPRE)
6	CYAN (BLEU CIEL)
7	BLANC

INK et PAPER sont des commandes "globales". Si vous spécifiez une nouvelle couleur d'ENCRE, la couleur de tout le texte sur l'écran change. Si vous spécifiez une nouvelle couleur de PAPIER, l'arrière-plan en entier prend cette couleur.

Faites attention à ne pas donner à INK et PAPER la même couleur. Si vous le faites vous ne verrez plus ce qui est imprimé sur l'écran.

Effacez la mémoire contenant le programme avec NEW et tapez ce programme :

```
10 FOR N=0 TO 7
20 CLS
30 PAPER N
40 INK 7-N
50 PRINT "PAPER" N, " INK"7-N
60 WAIT 200
70 NEXT
```

Vous n'avez pas à taper les espaces dans ce programme, PAPERN, WAIT200 et FORN=0TO7, marchent aussi bien. Les espaces (du moins, je l'espère) font que le programme est plus facile à lire et à comprendre.

Faites exécuter ce programme. Il montre huit différentes combinaisons de coloris de INK et PAPER. Bien plus de combinaisons sont possibles. Expérimentez jusqu'à ce que vous en ayez trouvé une qui vous convienne particulièrement. L'instruction PRINT à la ligne 50 sera expliquée dans le chapitre suivant.

Vous pouvez maintenant utiliser le clavier, entrer, éditer, lister, sauvegarder et charger des programmes, changer la couleur de l'encre et du papier, utiliser quelques-unes des instructions les plus communes de la machine. Ces dernières ont été choisies pour vous fournir des "outils" pour explorer et comprendre les caractéristiques les plus avancées de la machine.

Chapitre 6

L'AFFICHAGE DE L'ORIC

PRINT

L'instruction PRINT est utilisée pour "imprimer" des caractères (lettres ou nombres) sur l'écran de télévision et non (comme vous pourriez logiquement le supposer) sur l'imprimante papier.

D'elle-même, PRINT commence l'instruction sur l'écran à la première ligne libre. Elle peut être utilisée comme commande directe ou comme instruction dans un programme.

IMPRESSION DE NOMBRES

L'impression de nombres est très simple, par exemple :

```
PRINT 5
```

fait juste cela (à condition que vous appuyez sur RETURN). Le nombre apparaît au début de la première ligne libre, c'est-à-dire la première ligne depuis le haut de l'écran qui n'a pas déjà quelque chose d'imprimé dessus. Maintenant, essayez :

```
?6
```

Le nombre 6 devra apparaître au début de la ligne suivante.

Le point d'interrogation (?) est utilisé comme abréviation pour PRINT. Comme PRINT est utilisé dans un grand nombre de programmes, cette abréviation vous épargnera beaucoup de frappe au terminal.

Voici une chose intéressante - effacez la mémoire contenant le programme et entrez :

```
10?5
```

Listez le programme et vous obtiendrez :

```
10 PRINT5
```

ORIC vous laisse utiliser l'abréviation quand vous tapez des programmes, mais vous donne l'instruction non abrégée quand on LISTE ces programmes. Un bon exemple de la facilité d'usage.

J'utiliserai PRINT plutôt que ? tout au long de ce livre pour des motifs de clarté.

Vous avez aussi noté que je n'ai pas mis d'espace entre PRINT et 5 dans le dernier exemple, alors que je l'ai fait dans l'exemple précédent. L'espace est optionnel - PRINT 5 fait la même chose que PRINT5. De temps en temps l'insertion d'espaces comme dans le premier de ces exemples rend vos listings de programmes plus clairs.

Et si nous désirons imprimer un nombre quelque part dans le milieu de la ligne ? Essayons :

PRINT 6

C'est-à-dire en appuyant sur la barre d'espacement un certain nombre de fois avant de taper le 6. Après l'avoir entré vous trouverez que le 6 est imprimé exactement où il aurait été imprimé sans les espaces ! Ceci est dû au fait que la commande PRINT est conçue pour traiter le premier nombre rencontré et ignore les espaces. Nous verrons comment traiter cela bientôt.

IMPRESSION DE CARACTERES

Les caractères posent un problème différent. Par exemple si nous entrons

PRINT A

l'ordinateur imprime un 0 au commencement de la première ligne libre.

Cela signifie que l'ordinateur cherche un nombre et non une lettre. Il pense que "A" est une variable et comme elle n'a pas été initialisée précédemment, il fait l'hypothèse qu'elle a comme valeur zéro.

L'astuce est de mettre tous les caractères que nous voulons imprimer à l'intérieur de guillemets (""), par exemple

PRINT "A"

Le "A" apparaîtra, à nouveau au début de la première ligne vide.

CHAINES DE CARACTERES

Les caractères ou les nombres à l'intérieur de guillemets sont appelés des chaînes de caractères (STRINGS). Nous en apprendrons plus à leur sujet au chapitre 9.

Expérimentez avec différents messages à l'intérieur de guillemets. Effacez l'écran de manière à vous débarrasser de tout le fatras qui s'y est accumulé.

PRINT "UN MESSAGE A L'ECRAN"

PRINT " BEAUCOUP D'ESPACES "

En utilisant des espaces nous pouvons déplacer un message dans n'importe quel endroit de la ligne. Le message peut inclure des nombres mais pas comme partie d'un calcul. Essayez l'effet de :

PRINT "5678"

5678 est placé sur l'écran exactement à la même place que si "ABCD" avait été le message. Pour voir les différents effets clairement, essayez :

PRINT 5+6

et **PRINT "5+6"**

Sans guillements le programme calcule et affiche le résultat. Avec les guillemets, il affiche exactement ce qui est à l'intérieur des guillemets, ni plus ni moins. Ce que nous faisons instinctivement, ORIC doit être programmé pour le faire. Les guillements autour de n'importe quoi, que ce soient des nombres ou des lettres, disent au micro-ordinateur que c'est juste un message ou une étiquette équivalent à un numéro de maison ou à un nom qui doit être exactement imprimé tel quel. Nous pouvons inclure autant d'espaces que nous voulons dans un tel message. Un espace est traité de la même manière qu'une lettre ou qu'un nombre à l'intérieur, s'il est à l'intérieur de guillements.

SEPARATEURS DU PRINT

Nous pouvons aussi utiliser des espaces à l'intérieur des guillements pour placer le résultat d'un calcul où nous le voulons sur l'écran. Nous pouvons essayer :

PRINT " "

suivi par

PRINT 5+6

Mais les commandes PRINT successives mettent juste l'espace et le résultat sur des lignes successives.

POINT VIRGULE

Maintenant essayez :

PRINT " ";5+6

Le point virgule est l'un de ceux que l'on appelle les séparateurs du print, et ici il dit au micro-ordinateur que les deux éléments doivent se suivre l'un derrière l'autre directement. En général un point virgule indique que nous avons plus d'un élément qui suit PRINT et que ces éléments doivent être imprimés consécutivement (l'un derrière l'autre).

Afin de vous épargner du temps et de l'espace mémoire, ORIC rend optionnel l'usage du point virgule comme séparateur du print. Quand deux éléments sont sur la même ligne de programme, ORIC fait l'hypothèse qu'un point virgule est entre eux.

L'exception à ceci est quand un nombre en suit un autre. Dans ce cas ORIC requiert un point virgule pour indiquer où un nombre s'arrête et où le suivant commence.

Par exemple, comparez

PRINT 1;2;3 et PRINT 123

Quand les nombres sont imprimés à l'écran un espace est laissé après le nombre (avant et après le nombre sur la machine V1.1). Ce n'est pas le cas avec les chaînes de caractères.

Par exemple, essayez :

PRINT "1""2""3"1;2;3;"123" et PRINT "en";"semble"

Sivous souhaitez mettre des espaces entre les chaînes de caractères, alors vous devez inclure les espaces dans les chaînes ou utiliser des chaînes séparées contenant des espaces.

Si vous avez un nombre suivi par une chaîne de caractères, alors vous n'avez pas besoin de mettre un espace au début de la chaîne, car il y aura un espace qui suivra le nombre. Si au contraire vous avez une chaîne suivie par un nombre, cela requiert un espace à la fin de la chaîne de caractères dans la machine V1.0. La machine V1.1 met des espaces à la fois avant et après les nombres.

Essayez quelques exemples pour vous habituer à cette idée :

```
PRINT "ORIC";" ";1
PRINT 10"BOUTEILLES VERTES"
PRINT "SVP laissez ";3;"litres"
(dans la machine V1.1 PRINT "SVP laissez";3;"litres")
PRINT "5+6="5+6
```

Vous verrez d'après le troisième exemple qu'ORIC accepte les lettres minuscules dans les chaînes de caractères, bien qu'il ne les accepte pas comme commande (ou comme variable). CTRL T dépositionne le "capitals lock" et vous permet de taper des lettres minuscules.

Essayons une lettre représentant un nombre. Comme nous l'avons vu, une lettre (ou un groupe de lettres) égale à un nombre est appelée une "variable". Essayez :

```
A=10:PRINT A "BOUTEILLES VERTES"
A=10:PRINT "A BOUTEILLES VERTES"
```

La première ligne devrait avoir pour résultat

```
10 BOUTEILLES VERTES
```

et la seconde donnera le résultat stupide

```
A BOUTEILLES VERTES
```

Cela confirme que le "A" entre guillemets est traité juste comme une lettre qui doit être imprimée et non considérée comme un nombre.

Le point virgule peut être aussi utilisé à la fin de l'instruction PRINT. Cela inhibe le retour chariot de telle manière que le prochain caractère imprimé se trouve au premier endroit libre dans la ligne courante de l'écran, au lieu du début de la ligne suivante.

Cela paraît compliqué, mais un exemple le rendra clair.

Effacez le programme mémoire et tapez :

```
10 CLS
20 PRINT:PRINT:PRINT:PRINT
30 PRINT "LE COMPTEUR MARQUE";
40 FOR N=1 TO 9
50 PRINT N;
60 WAIT 100
70 NEXT
```

Faites exécuter ce programme avec et sans les points virgules à la fin des lignes 30 et 50. Si vous avez la machine V1.1 vous n'avez pas besoin d'espaces comme dernier caractère de la chaîne de caractères de la ligne 30.

La ligne 20 dans ce programme est intéressante. L'instruction **PRINT** en elle-même n'imprime rien sur l'écran, mais déplace le début de ce qui doit être imprimé à l'écran à la ligne suivante. Pour démontrer ceci, essayez de modifier le nombre d'instructions **PRINT** de la ligne 20 en vous rappellant de les séparer par deux points (:). Essayez l'effet produit par la suppression de la ligne 20.

VIRGULE

La virgule (,) peut être utilisée comme séparateur dans l'instruction print.

L'effet de ce séparateur du print dépend de la version de la machine que vous possédez.

Dans la machine V1.0

La virgule insère trois espaces entre les éléments imprimés.

Toutefois, rappelez-vous que quand un nombre est imprimé, un espace est inséré par la virgule, ce qui donne une totalité de quatre espaces imprimés entre les nombres séparés par des virgules.

Les chaînes de caractères n'ont pas d'espaces supplémentaires insérés après elles. Aussi une virgule met trois espaces entre les chaînes de caractères.

Le nombre d'espaces entre éléments n'est pas affecté par la longueur de l'élément.

Si plusieurs virgules sont utilisées ensemble, chaque virgule additionnelle ajoute deux autres espaces.

Dans la machine V1.1

La virgule divise l'écran en colonnes, chacune ayant huit caractères de large. Les caractères ne sont pas normalement imprimés dans les deux premières positions horizontales (0 et 1) si bien que la virgule séparatrice commence l'impression des éléments aux positions 2, 10, 18, etc...

Les nombres ont des espaces devant eux. Si vous utilisez la virgule comme séparateur avec des nombres, alors l'espace précédent chaque nombre est imprimé au début de la position de la colonne correspondante. Aussi les nombres eux-mêmes commencent respectivement aux positions 3, 11, 19, etc...

Le nombre d'espaces entre les éléments est affecté par le nombre de caractères dans les éléments, car chaque élément commencera à la première colonne libre. Si un élément a plus de huit caractères de large (incluant les espaces précédents et suivant les éléments numériques) alors il prendra deux colonnes ou plus. Si plusieurs virgules sont utilisées successivement, chaque virgule additionnelle produit une colonne pleine d'espaces.

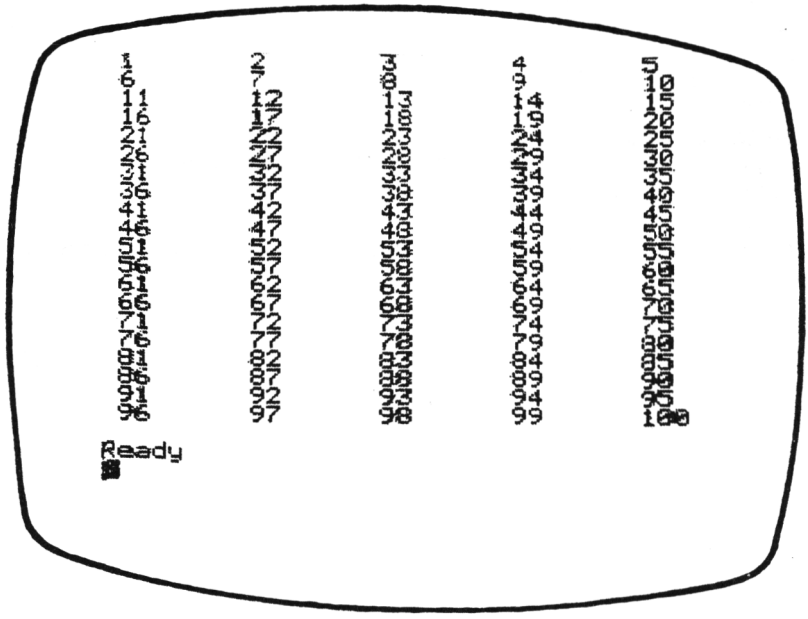
Quelque soit la version de machine que vous possédez, essayez ces exemples :

```
PRINT 1,2,3
PRINT "A","B","C"
PRINT "A",1,"B"
PRINT 1684,3,18
PRINT "MAINS","GENOUX","ET","RENE","MARGUERITE"
PRINT 1,,2,,3
```

La virgule, comme le point virgule, peut être placée à la fin de l'instruction PRINT.

Effacez la mémoire contenant le programme, tapez et exécutez :

```
10 CLS
20 FOR N=1 TO 100
30 PRINT N,
40 NEXT
```



Le résultat obtenu par ce programme sur la machine V1.1 est montré ci-dessus. Vous pouvez faire exécuter ce programme sur la machine V1.0 mais vous n'obtiendrez pas de résultat aussi nettement "tabulé".

Bien que la virgule utilisée comme séparateur sur la machine V1.1 permette de mettre des éléments dans les colonnes, elle ne nous donne pas de contrôle sur l'emplacement où se trouve ces colonnes. Le séparateur TAB -qui ne fonctionne correctement que sur la machine V1.1, est beaucoup plus flexible.

```
PRINT TAB(n)...
```

fait que les éléments suivant TAB(n), qu'ils soient des mots ou des nombres, commencent à imprimer n espaces après le début de la ligne. Ceci est utilisé pour créer des tables d'informations en colonnes. Nous pouvons utiliser plusieurs TAB sur une même ligne à condition que chaque valeur de n soit plus grande que la précédente - le programme calcule l'ampleur du déplacement nécessaire pour se rendre à la prochaine valeur indiquée par TAB. Par exemple :

```
PRINT TAB(5)"PREMIER"TAB(15)"SECOND"
```

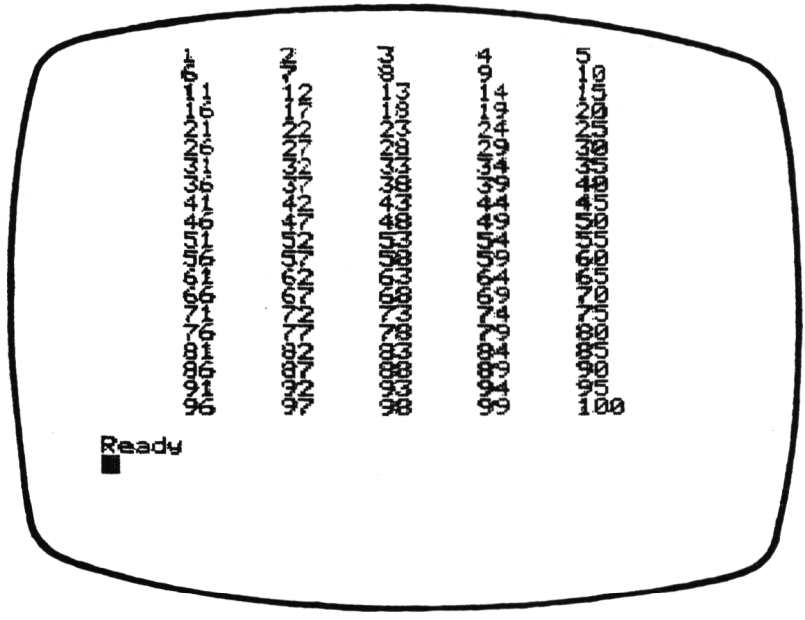
imprimera le mot PREMIER en commençant à la cinquième position et le mot SECOND à la quinzième. Essayez de changer les positions indiquées pour en voir l'effet. Par exemple, essayez d'imprimer le mot SECOND en commençant à la position 13 et ensuite à la position 12.

Vous verrez qu'il ne peut imprimer sur la même ligne que si la nouvelle position indiquée par TAB est plus loin que le dernier caractère imprimé, c'est-à-dire avec 7 caractères dans le mot PREMIER la lettre R est imprimée à la position 12. Essayer d'imprimer un autre mot ou nombre en commençant à la position 12 altérerait l'original, aussi le mot SECOND est imprimé à la position 12 mais sur la ligne suivante.

Manifestement nos TAB doivent être en ordre croissant, nous ne pouvons pas tabuler en arrière.

Effacez la mémoire contenant le programme et tapez ce programme. Comparez les résultats obtenus avec celui de l'exemple du programme précédent. Ne vous souciez pas de "STEP" dans la ligne 20. Nous verrons cela au chapitre 8.

```
10 CLS
20 FOR N=1 TO 100 STEP 5
30 PRINT
TAB(6)N;TAB(12)N+1;TAB(18)N+2;TAB(24)N+3;TAB(30)N+4
40 NEXT
```



SPC

SPC (abréviation de SPACE) est un séparateur utile.

PRINT SPC(n)...

insérera n espaces avant l'élément à imprimer.

n doit être un nombre positif entier pas plus grand que 255.

Vous pouvez placer un message partout sur l'écran en utilisant SPC. Par exemple, essayez :

```
CLS:PRINT SPC(255)SPC(249)"MILIEU DE L'ECRAN"
```

Ce n'est toutefois pas une bonne technique. C'est plutôt lent et toute autre impression déjà sur l'écran est perdue, car CLS est requis pour positionner le curseur dans le coin supérieur gauche de l'écran.

SPC peut être utilisé de manière plus utile entre deux éléments à imprimer. Essayons par exemple :

```
PRINT "DIX"SPC(10)"ESPACES"
```

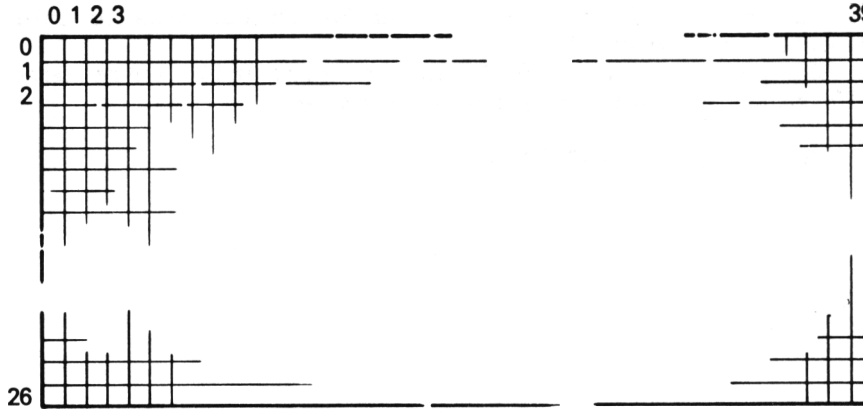
PRINT @

PRINT @ vous permet de spécifier la position sur l'écran d'un message imprimé qui doit commencer à telle colonne de l'écran (de 0 à 39) et à telle rangée de l'écran (de 0 à 25). Cette instruction n'est disponible que sur la machine V1.1.

Le format est

```
PRINT @ X,Y;...
```

où X spécifie le numéro de colonne et où Y spécifie le numéro de rangée comme montré ci-dessous.



Ainsi,

```
PRINT @ 4,3;"ORIC"
PRINT @ 5,5;1
```

donnerait

	0	1	2	3	4	5	6	7		
0										
1										
2										
3					O		I	C		
4										
5						1				

L'espace entre PRINT et @ est optionnel, tout comme l'est l'espace entre @ et le numéro de colonne. Le point virgule séparant le numéro de rangée du premier élément à imprimer est obligatoire. Si vous ne le mettez pas, vous obtiendrez une erreur de syntaxe.

Vous pouvez utiliser l'instruction PRINT @ pour imprimer sur les colonnes 0 et 1 à condition que vous spécifiiez que le papier soit noir et l'encre blanche. Toute autre combinaison d'encre et de papier vous donnerait des effets tout-à-fait bizarres et il en résulterait la perte d'une partie de votre message. Il est prudent de restreindre vos impressions aux colonnes 2 à 39.

Nous verrons pourquoi lorsque nous regarderons les attributs au prochain chapitre.

Nous pouvons utiliser plusieurs types de séparateurs dans la même instruction PRINT.

Essayez :

```
PRINT SPC(5)1,2;"LACE MON SOULIER"
PRINT @ 6,7;"HELLO"TAB(20)"DOLLY"
```

(Le deuxième exemple ne fonctionnera que sur la machine V1.1.)

ARITHMETIQUE

L'ordinateur est utilisé pour traiter de l'information. Jusqu'à maintenant, nous avons étudié principalement les mots et où ils doivent être imprimés. Nous avons aussi glissé occasionnellement une addition, par exemple :

```
PRINT 5+6
```

donne la réponse 11.

Nous pouvons faire des additions, des soustractions, des multiplications et des divisions à condition que nous nous habituions à quelques symboles peu familiers. Le clavier dispose des signes + et - mais non pas des signes conventionnels x et : . Au lieu de cela nous utiliserons étoile (*) et slash (/) si bien que 3 plus 4 multipliés par 5, divisé par 6 est écrit :

```
3+4*5/6
```

L'ordre de calcul est important. La multiplication et la division sont exécutées d'abord, l'addition et la soustraction ensuite, si bien que la formule ci-dessus est calculée comme suit :

```
3+((4*5)/6)
```

Nous pouvons utiliser ces calculs, ainsi que beaucoup de fonctions plus avancées expliquées aux chapitres 8 et 9, comme élément d'à peu près n'importe quelle ligne de programme, y compris les instructions PRINT.

Chapitre 7

QUELQUES CARACTERES BIZARRES

CHR$

Chaque caractère que vous pouvez taper à partir du clavier et afficher à l'écran a un nombre qui lui est associé. Ces nombres sont connus comme les codes ASCII, et sont listés en annexe A.

CHR$ convertit un nombre en une chaîne de caractères contenant le caractère associé à ce nombre. Par exemple, le code ASCII pour A est 65. Ainsi :

```
PRINT CHR$ (65)
```

donne le même résultat que

```
PRINT "A"
```

CARACTERES DE CONTROLE

Si vous regardez l'annexe A vous verrez que les codes ASCII commencent à 32. Les codes inférieurs à 32 ne correspondent à aucun caractère sur l'écran. De tels codes produisent des caractères connus sous le nom de **caractères de contrôle.**

Par exemple, essayez :

```
PRINT CHR$(4)"DOUBLE"
```

Cela vous donnera une duplication de l'impression de la même manière que CTRL D l'a fait au chapitre 2.
Utilisez CTRL D pour revenir à l'impression simple.
Vous avez peut-être remarqué que D est la quatrième lettre de l'alphabet et que CTRL D a fait la même chose que PRINT CHR$(4). Est-ce une coïncidence ? Essayons de trouver.
Vous vous souviendrez que CTRL G a causé un "ping". G est la septième lettre de l'alphabet; aussi, essayons

```
PRINT CHR$(7)
```

Un test de plus pour les sceptiques. CTRL Q cache le curseur. Q est la dix-septième lettre de l'alphabet; aussi, essayons

```
PRINT CHR$(17)
```

Utilisez soit CTRL Q ou PRINT CHR$(17) pour obtenir à nouveau le curseur.

Voici la liste des caractères de contrôle que vous pourrez trouver utile pour vos programmes. Certains appareils périphériques (tels certains types d'imprimantes) utilisent d'autres codes de contrôle qui seront listés dans les documentations qui accompagnent ces appareils.

CHR$(1) copie un caractère de l'écran dans le tampon du clavier.
CHR$(4) permet la duplication de l'impression (bascule).
CHR$(6) cliquetis du clavier (bascule).
CHR$(7) cause un "ping".
CHR$(8) déplace le curseur d'un espace vers la gauche.
CHR$(9) déplace le curseur d'un espace vers la droite.
CHR$(10) descend le curseur d'une ligne.
CHR$(11) monte le curseur d'une ligne.
CHR$(12) efface l'écran.
CHR$(13) simule l'enfoncement de la touche RETURN.
CHR$(17) cache le curseur (bascule).
CHR$(19) supprime la sortie des résultats à l'écran (bascule).
CHR$(20) contrôle le blocage des lettres majuscules (bascule).
CHR$(29) permet l'impression dans les colonnes d'extrême gauche (bascule).

Les caractères de contrôle doivent être précédés par une instruction telle que PRINT.

Certains caractères de contrôle sont plus utiles que d'autres. Vous trouverez probablement que les codes 4,8 à 12 et 17 sont les plus courants, mais les autres sont disponibles si vous désirez les utiliser.

ESCAPE

Il y a un autre caractère de contrôle que je n'ai pas mis sur la liste générale, mais il est si important qu'il mérite une section à lui tout seul.

C'est CHR$(27) qui simule l'enfoncement de la touche ESCAPE et qui donc nous permet de mettre des attributs à l'écran sous le contrôle du programme.

Nous avons vu au chapitre 2 comment les attributs pouvaient être utilisés pour générer des affichages à l'écran plutôt inhabituels (et plutôt spectaculaires). CHR$(27) nous permet de faire la même chose mais maintenant nous pouvons utiliser des programmes pour générer ces affichages automatiquement, les changer et les déplacer.

Le caractère suivant CHR$(27) n'est pas imprimé mais au lieu de cela détermine l'attribut qui affecte tous les caractères qui le suivent jusqu'à ce que l'on rencontre, soit la fin de la ligne, soit un second attribut qui annule l'effet du premier.

Quand vous préparez un affichage à l'écran, il est important de se souvenir que les attributs sont imprimés comme des caractères vides et que chacun d'entre eux occupe la position d'un caractère sur l'écran.

Listons d'abord les essais des différents caractères qui peuvent suivre CHR$(27). Nous verrons ensuite quelques exemples d'utilisation de ceux-ci.

```
PRINT CHR$(27) " " - encre noire
      "        "A" - encre rouge
      "        "B" - encre verte
      "        "C" - encre jaune
      "        "D" - encre bleue
      "        "E" - encre magenta
      "        "F" - encre cyan
      "        "G" - encre blanche
      "        "H" - standard, simple hauteur,
                     permanent.
      "        "I" - alternatif, simple hauteur,
                     permanent.
      "        "J" - standard, double hauteur,
                     permanent
      "        "K" - alternatif, double hauteur,
                     permanent
      "        "L" - standard, simple hauteur,
                     clignotant
      "        "M" - alternatif, simple hauteur,
                     clignotant
      "        "N" - standard, double hauteur,
                     clignotant
      "        "O" - alternatif, double hauteur,
                     clignotant
      "        "P" - papier noir
      "        "Q" - papier rouge
      "        "R" - papier vert
      "        "S" - papier jaune
      "        "T" - papier bleu
      "        "U" - papier magenta
      "        "V" - papier cyan
      "        "W" - papier blanc.
```

Essayons d'abord quelques exemples pour vous accoutumer à cette idée :

```
PRINT CHR$(27)"A"CHR$(27)"T"SPC(12)"ROUGE SUR BLEU"
PRINT CHR$(27)"E"CHR$(27)"LEXEMPLE CLIGNOTANT"
PRINT CHR$(27)"C"CHR$(27)"IQUELQUES CARACTERES ETRANGES"
```

Faites-en quelques autres pour vous-même. Effacez l'écran occasionnellement de telle manière que vous ne soyez pas complètement perdu.

Un programme, qui peut être répété, est habituellement plus intéressant que des commandes immédiates. Aussi, essayez :

```
10 PAPIER0:INK 3'PAPIER NOIR ENCRE JAUNE
20 PRINTCHR$(12)CHR$(4)CHR$(17)
30 REM EFFACE L'ECRAN, DOUBLE HAUTEUR CACHE LE CURSEUR
40 PRINTSPC(9)CHR$(27)"JGROSSES LETTRES JAUNES"
50 PRINT:PRINT:PRINT
60 PRINTSPC(10)CHR$(27)"T"CHR$(27)
70 PRINT:PRINT:PRINT
80 PRINTSPC(9)CHR$(27)"A"CHR$(27)"V"CHR$(27)"JAUSSI
   ROUGE SUR CYAN"CHR$(27)"P"
90 GOTO 90
```

Ce programme a un certain nombre de points intéressants.

Premièrement, j'ai choisi la couleur du papier en noir. Si vous ne spécifiez pas une couleur en arrière plan lorsque vous contrôlez des affichages avec des attributs, alors votre arrière plan pour cette ligne sera noir par défaut. Mettre la couleur de l'arrière-plan en général en noir gagne du code (ligne d'instruction générée par la machine) et donne habituellement des affichages plus nets. Essayez de changer la ligne 10 en PAPER7:INK 3, et vous verrez ce que je veux dire.

Parce que j'ai spécifié jaune comme la couleur d'encre globale, toutes les impressions seront en jaune jusqu'à ce que je change la couleur de l'encre sur une ligne spécifique en utilisant un attribut, comme je l'ai fait à la ligne 80.

J'ai utilisé PRINT CHR$(12) plutôt que CLS pour effacer l'écran. Cette instruction non seulement efface l'écran mais aussi déplace le curseur d'une ligne vers le bas (de la ligne 0 de l'écran à la ligne 1). Comme nous l'avons vu au chapitre 2, nous avons besoin de faire ceci pour que l'impression en double hauteur fonctionne correctement.

La ligne de programme 50 déplace le curseur de la ligne de l'écran 1 à la ligne 5 (saute donc les lignes 2,3 et 4). Comme la ligne 5 est une ligne de numéro impair à l'écran, la double impression fonctionnera correctement ici. En général l'impression de lignes en double hauteur nécessite d'avoir un double espace (ou un multiple de celui-ci) entre elles, ou encore pas d'espace du tout. Essayez le résultat produit en changeant les lignes 50 et 70 de telle manière que chacune contienne 5 instructions PRINT. Changez-les de nouveau de telle manière qu'elles contiennent seulement une instruction PRINT.

Finalement la ligne 90 met le programme en boucle continuelle de telle manière que le mot "Ready" n'apparaisse pas et ne gâche pas l'affichage à l'écran. Sortez de cette boucle par CTRL C. N'oubliez pas de restaurer le curseur (CTRL Q) et de supprimer l'impression double (CTRL D) après avoir arrêté le programme.

Jouez avec ce programme et utilisez-le comme un modèle pour écrire des programmes similaires pour vous-mêmes.

Tapez PAPER 7:INK 0:CLS, cela efface l'écran et repositionnera l'encre en noir et le papier en blanc.

Voici un autre programme que vous pourrez trouver intéressant. Essayez de trouver ce qu'il fait.

```
10 PAPER 0:INK 2
20 PRINTCHR$(12)CHR$(17)
30 FOR N = 1 TO 21
40 PRINTCHR$(4)SPC(N)CHR$(27)"KNS£S£S£S£,W[^/^"CHR$(10)
50 PRINTCHR$(4)SPC(N+1)"o"SPC(3)"o"SPC(3)"oooo";
   CHR$(11)CHR$(11)CHR$(11)
60 WAIT 10
70 NEXT
80 PRINTCHR$(17)
```

PLOT

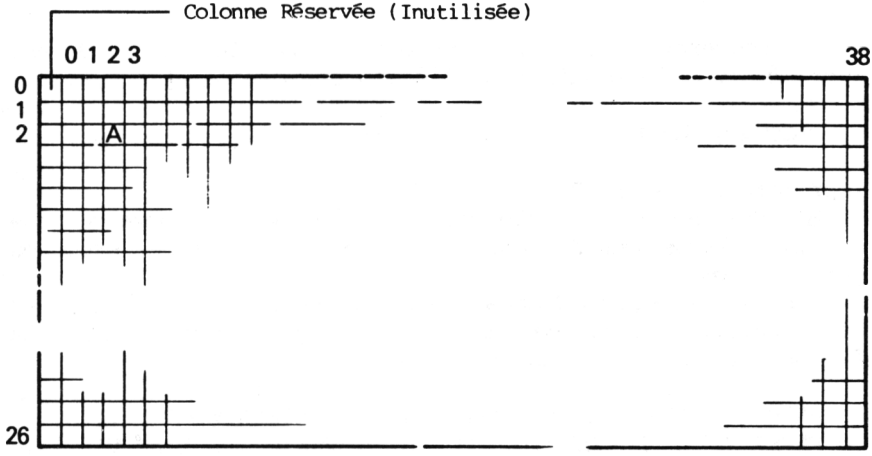
Vous pouvez positionner un message n'importe où sur l'écran en utilisant SPC ou TAB avec l'instruction PRINT. Toutefois cela peut prendre pas mal de travail. Ce qui est nécessaire ici, c'est une instruction qui imprime sur l'écran en se positionnant à partir d'un certain nombre d'espaces comptés à partir de la gauche de l'écran, et d'un certain nombre de lignes comptées à partir du haut de l'écran.

Si vous avez la machine V1.1 vous pouvez faire ceci avec PRINT @. Toutefois l'instruction PLOT fonctionne avec les deux versions de la machine. Le format est :

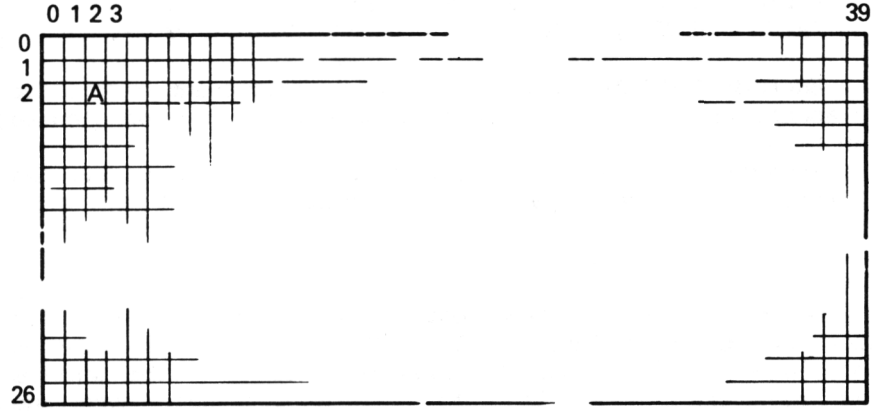
```
PLOT X,Y,"..."
```

où X est le nombre d'espaces à partir de la gauche de l'écran et où Y est le nombre de lignes à partir du haut de l'écran.

Le nombre de lignes et de colonnes est légèrement différent pour chaque version :



ORIC (V1.0)



ORIC ATMOS (V1.1)

Par exemple, PLOT 2,2"A" imprimera la lettre A dans les positions montrées ci-dessus.

PLOT n'est normalement utilisé qu'avec des chaînes de caractères. Si vous essayez :

```
PLOT 6,6,65
```

vous trouverez qu'au lieu du nombre 65, la lettre A est imprimée dans la position de l'écran correspondant à 6,6.

```
PLOT 20,14,90
```

imprimera la lettre Z.

L'instruction PLOT interprète le 3ème nombre comme un code ASCII. Référez-vous à l'annexe A et essayez quelques exemples par vous-même.

Dans les exemples donnés ci-dessus je n'ai pas utilisé les deux colonnes de gauche. Regardons ce qui arrive si nous le faisons. Effacez l'écran et entrez :

```
PAPER 7:INK 4:PLOT 0,0,"EN HAUT A GAUCHE"(Machine V1.0)
ou PAPER 7:INK 4:PLOT 1,0,"EN HAUT A GAUCHE"(Machine V1.1).
```

Pas très inspirant. Effacez l'écran de nouveau et entrez :

```
PAPER 0:PLOT 0,0,"EN HAUT A GAUCHE" (Machine V1.0)
ou PAPER 0:PLOT 1,0,"EN HAUT A GAUCHE" (Machine V1.1).
```

EN HAUT A GAUCHE devrait apparaître maintenant en couleur blanche. Bien que vous ayez positionné la couleur de l'encre au bleu.

Essayez de changer le message de telle manière que les lettres sur l'écran apparaissent en bleu. Entrez

```
INK 4
```

Le message devient bleu, mais vous avez perdu la première lettre. On lit maintenant EN HAUT A GAUCHE. Les messages placés sur l'écran par la commande PLOT de telle manière qu'ils commencent à la colonne 0 apparaissent avec une couleur d'encre blanche. Si vous essayez de changer la couleur de l'encre vous perdez le premier caractère du message. Ceci est dû au fait que la seconde colonne à partir de la gauche est utilisée pour contenir les attributs contrôlant la couleur de l'avant-plan (encre) de l'affichage à l'écran.

Si vous avez la machine V1.1 vous pouvez aussi utiliser la commande PLOT dans la colonne de l'extrême gauche. Si vous le faites, l'arrière-plan de la rangée toute entière devient noir et le message apparaît avec une encre blanche.

LORES 0

Jusqu'à maintenant vous avez utilisé le mode TEXT pour afficher les messages à l'écran. La machine fonctionne automatiquement en mode TEXT quand vous vous connectez ou lorsque vous appuyez sur l'interrupteur RESET. Toutefois il y a un autre mode qui vous permet d'utiliser la colonne 0 pour vos messages en positionnant par défaut les couleurs du papier à noir et de l'encre à blanc de telle manière que la colonne 0 n'est pas requise pour des attributs. C'est le mode de basse résolution (LOW RESOLUTION) 0 ou en abrégé LORES 0.

Ce mode est activé en utilisant l'instruction LORES 0 aussi bien dans un programme que dans une commande directe.

LORES 0 est aussi une instruction qui efface l'écran dans ce mode. Si vous utilisez CLS ou CTRL L la machine revient au mode TEXT. Si vous utilisez une instruction INK pour changer la couleur de l'avant-plan vous perdrez toutes les impressions dans la colonne 0. Essayer de changer la couleur de l'arrière-plan avec une instruction PAPER produit un résultat bizarre. Essayez-le et regardez le résultat.

L'écran ne peut pas être défilé dans le mode LORES 0. Si vous essayez d'imprimer au-delà de la ligne 26 l'écran redéfile automatiquement en mode TEXT.

Dans le mode LORES 0 les instructions PRINT sont utilisées de manière normale.

L'instruction TEXT peut être utilisée pour revenir en mode TEXT. Toutefois ceci n'efface pas l'écran ni ne restaure pas les couleurs normales de papier. CLS fait tout cela d'un coup.

ATTRIBUTS DE L'INSTRUCTION PLOT

Nous avons vu qu'afficher un nombre à l'aide de l'instruction PLOT a pour résultat que ce nombre est traité comme un code ASCII. Le plus petit code ASCII étant 32 qu'advient-il si nous utilisons la fonction PLOT avec un nombre inférieur à 32 ?

La réponse est qu'utiliser la fonction PLOT avec des nombres inférieurs à 32 constitue un autre moyen pour positionner des attributs sur l'écran et c'est une méthode très pratique, parce qu'elle permet de spécifier la position où vous souhaitez que l'attribut soit placé.

```
PLOT X,Y,0  - encre noire
 "       1  - encre rouge
 "       2  - encre verte
 "       3  - encre jaune
 "       4  - encre bleue
 "       5  - encre magenta
 "       6  - encre cyan
 "       7  - encre blanche
 "       8  - standard, simple hauteur, constant
 "       9  - alternatif, simple hauteur, constant
 "       10 - standard, double hauteur, constant
 "       11 - alternatif, double hauteur, constant
 "       12 - standard, simple hauteur, clignotant
 "       13 - alternatif, simple hauteur, clignotant
 "       14 - standard, double hauteur, clignotant
 "       15 - alternatif, double hauteur, clignotant
 "       16 - papier noir
 "       17 - papier rouge
 "       18 - papier vert
 "       19 - papier jaune
 "       20 - papier bleu
 "       21 - papier magenta
 "       22 - papier cyan
 "       23 - papier blanc
 "       24 à 31 - peuvent affecter la synchronisation
de l'écran
```

Je ne vous conseille pas d'utiliser l'instruction PLOT avec les nombres 24 à 31. Certains des effets produits peuvent ne pas être très bons pour votre téléviseur.

Le prochain programme montre comment les attributs utilisés dans l'instruction PLOT peuvent affecter un affichage sur l'écran. Voyez si vous comprendrez ce qu'il fait. Arrêtez-le avec CTRL C et obtenez de nouveau le curseur avec CTRL Q.

```
10 PRINT CHR$(12)CHR$(17)
20 PAPER 7:INK 0
30 PLOT5,11,"VOICI UN NOM EN COULEURS"
40 PLOT5,12,"VOICI UN NOM EN COULEURS"
50 WAIT 150
60 PLOT2,11,10:PLOT2,12,10
70 WAIT 50
80 FOR N=1 TO 6
90 PLOT11,11,N:PLOT16,11,7-N:PLOT21,11,0
100 PLOT11,12,7-N:PLOT16,N:PLOT21,12,0
110 WAIT 50
120 NEXT
130 PLOT11,11,0:PLOT11,12,0
140 PLOT16,11,0:PLOT16,12,0
150 WAIT 50
160 FOR N=1 TO 6
170 PLOT11,12,23-N:PLOT16,12,16+N:PLOT21,11,23
180 PLOT11,12,23+N:PLOT16,12,16+N:PLOT21,12,23
190 WAIT 50
200 NEXT
210 PLOT11,11,23:PLOT11,12,23
220 PLOT16,11,23:PLOT16,12,23
230 GOTO 70
```

ATTRIBUTS AVEC PRINT @

Si vous avez la machine V1.1, il existe encore une méthode pour mettre les attributs à l'endroit sélectionné sur l'écran. C'est d'utiliser l'instruction PRINT @ suivi de CHR$(27).

Essayez :

```
CLS:PRINT@9,12,CHR$(27)"A"CHR$(27)"TROUGE SUR BLEU"
```

INPUT

L'instruction INPUT est utilisée pour obtenir de l'information à partir du clavier, pendant que le programme est en fonctionnement.

Tapez et faites exécuter le programme suivant :

```
10 INPUT "QUEL NOMBRE";N
20 PRINT N,10*N
30 GOTO 10
```

La ligne 10 contient l'instruction INPUT. Elle imprimera sur la première ligne libre de l'écran tout ce qui se trouve entre les deux doubles guillemets. Puis ellè attendra que vous tapiez un nombre, suivi par la touche RETURN.

N peut être positif ou négatif. Il n'a pas besoin d'être un nombre entier. L'instruction INPUT met un espace et un point d'interrogation après le message. L'instruction INPUT fonctionne de la même manière si vous ne lui mettez pas de message. Alors seul le point d'interrogation (caractère de sollicitation, en anglais "prompt") apparaît à l'écran. Essayez en changeant la ligne 10 en :

```
10 INPUT N
```

Si vous voulez être certain que seuls des nombres entiers soient entrés, mettrez un signe % après le nom de la variable. Alors tout nombre entré sera arrondi à la valeur de l'entier immédiatement inférieur.

Essayez de changer les lignes 10 et 20 en :

```
10 INPUT N%
20 PRINT N%
```

Si vous entrez des lettres alors qu'ORIC attend un nombre, ORIC vous demandera de rentrer de nouveau l'information correcte en vous donnant le message d'erreur REDO FROM START.

Une deuxième forme de l'instruction INPUT vous permet d'entrer à la fois des lettres et des nombres. Cette forme de l'instruction met l'information entrée entre guillemets et la transforme ainsi en chaîne de caractères (voir le début de ce chapitre). Que l'information rentrée soit des lettres ou des nombres, cela fonctionne.

Pour entrer des informations dans les chaînes de caractères, le format est

INPUT"...";A$ ou simplement INPUT A$

Notez le signe $. Cela dénote une chaîne de caractères.

Le prochain programme démontre l'usage de ce format d'instructions :

```
10 INPUT "QUEL EST VOTRE NOM";A$
20 PRINT "BONJOUR" A$
```

```
 RUN
QUEL EST VOTRE NOM? ALAIN
BONJOUR ALAIN

Ready
```

Souvenez-vous que pour les deux formats de l'instruction INPUT, vous devez appuyer sur la touche RETURN après avoir entré vos données.

Si vous avez un message dans votre instruction INPUT vous devez avoir un point virgule comme séparateur entre le message et les variables de l'instruction INPUT. Si vous n'avez pas de message, vous n'avez pas besoin d'un point virgule.

Chapitre 8

IL FAUT PRENDRE UNE DECISION

DECISIONS

La plupart des programmes que nous avons vu jusqu'à maintenant s'exécutaient du début jusqu'à la fin. C'est inhabituel, car l'ordinateur est un très bon preneur de décision. Nous pouvons faire en sorte que l'ordinateur décide d'exécuter ou non une commande; ou encore lui faire exécuter des lignes dans un ordre différent, en fonction de ce qui s'est passé précédemment. Quelques exemples sont plus utiles que des mots.

IF...THEN

Examinons la commande IF...THEN. Elle est utilisée comme elle est écrite. Si (IF) quelque chose est vrai alors (THEN) il faut faire quelque chose. Voici un exemple :

```
IF X=5 THEN PRINT "X=5"
```

Ce qui vient après le THEN doit être une commande et ce qui vient après le IF doit être une sorte d'expression qui est soit vraie, soit fausse.

L'ORIC comprend un certain nombre de symboles dans une instruction IF et ceci inclut :

= égal
< inférieur
> supérieur
<= plus petit ou égal (ou =<)
>= plus grand ou égal (ou =>)
<> différent

Ce programme montre quelques-unes de ces intructions en action.

```
10 INPUT"QUEL NOMBRE";N
20 IF N=5 THEN PRINT"VOUS AVEZ TAPE 5"
30 IF N<0 THEN PRINT"VOTRE NOMBRE EST NEGATIF"
40 IF N<>8 THEN PRINT"CE N'ETAIT PAS 8"
50 IF N<=1 THEN GOTO 10
60 IF N>10 THEN STOP
70 GOTO 10
```

Essayez ce programme et voyez si vous comprenez ce qu'il fait. Remarquez que vous n'avez pas besoin d'utiliser les instructions PRINT. Vous pouvez avoir des GOTO ou des STOP ou n'importe quoi dans une instruction IF.

Examinez par exemple ce stupide programme :

```
10 INPUT "QUEL NOMBRE";N
20 IF N>5 THEN N=32
30 PRINT "VOUS AVEZ ENTRE" N
40 GOTO 10
```

Ce programme donne une réponse correcte seulement si votre nombre est inférieur ou égal à 5.

AND, OR, NOT

Examinez ce programme. Il fait les commentaires sur les âges inférieurs à 2, supérieurs à 110 et compris entre 13 et 19. Ainsi un certain nombre de décisions peuvent être prises à propos d'une variable.

```
10 INPUT "QUEL EST VOTRE AGE";A
20 IF A<2 THEN PRINT "JE NE VOUS CROIS PAS"
30 IF A>110 THEN PRINT "VOUS DEVEZ ETRE UN EXTRA
   TERRESTRE
40 IF A>12 AND A<20 THEN PRINT "VOUS ETES UN ADOLESCENT"
50 GOTO 10
```

A la ligne 40 notez comment j'ai utilisé le mot AND (et, intersection) pour faire prendre une décision plus compliquée par l'ORIC. Nous pouvons utiliser d'autres mots pour assembler des instructions IF. Ceux-ci sont OR (OU, union) et NOT (négation, complémentaire). Ainsi, je pourrai écrire :

```
IF X<10 AND A=5 OR Y=X THEN GOTO 20
```

Cela signifie d'aller à la ligne 20 si à la fois X est inférieur à 10 et que A est égal à 5 ou Y est égal à X. Ceci est très compliqué et n'est donné qu'à titre d'exemple. L'usage de décisions multiples vient avec la pratique et il est suffisant à ce stade de savoir qu'elles sont disponibles afin de pouvoir les utiliser ultérieurement.

NOT est aussi utilisé dans les décisions :

```
IF NOT A=B THEN...
```

est identique à

```
IF A<>B THEN...
```

ou en français

```
SI A N'EST PAS EGAL A B ALORS...
```

IF...THEN...ELSE

L'instruction IF...THEN...ELSE (si...alors...sinon) est une extension de IF...THEN mais donne des décisions plus puissantes et plus nettes. Elle teste si une condition ou un nombre de conditions sont vraies. Si elles le sont elle exécute un certain nombre d'opérations, sinon elle exécute un autre ensemble d'opérations.

Par exemple les lignes 40 et 50 du dernier programme peuvent être réunies dans une seule ligne :

```
40 IF A>12 AND A<20 THEN PRINT "VOUS ETES UN ADOLESCENT"
   ELSE GOTO 10
```

GOTO, lorsqu'il est utilisé après THEN ou ELSE peut être omis, si bien que dans cet exemple nous pouvons avoir ELSE 10 au lieu de ELSE GOTO 10.

ATTENTION - Les ordinateurs comme ORIC, sont plus logiques que nous le sommes. Ils font les choses exactement comme on leur dit. L'instruction IF est la source de bien des erreurs dans les programmmes.

Etudiez ce programme :

```
10 INPUT"QUEL NOMBRE";N
20 IF N<5 THEN PRINT"N<5"
30 IF N>5 THEN PRINT"N>5"
```

Nous avons oublié de dire à ORIC ce qu'il fallait faire si N était égal à 5. Dans ce cas il ne fera rien, mais dans les programmes plus longs ce type d'erreur peut être désastreux.

La structure IF...THEN...ELSE est moins sujette à ce type d'erreur puisque ELSE couvre toutes les conditions non spécifiées par le IF. Le programme devient :

```
10 INPUT"QUEL NOMBRE;"N
20 IF N<5 THEN PRINT"N<5" ELSE PRINT "N>=5"
```

L'expression OR (ou) est vraie quand l'une ou l'autre ou les deux conditions sont satisfaites :

```
10 INPUT"PREMIER NOMBRE";X
20 INPUT"SECOND NOMBRE";Y
30 IF X=5 OR Y=6 THEN PRINT "BINGO" ELSE 10
```

Notez comment l'instruction PRINT s'exécute quand X=5 et Y=6 sont vrais ou les deux à la fois.

Si nous le souhaitons nous pouvons utiliser l'instruction IF pour faire une boucle de programme de manière à exécuter une partie du programme un certain nombre de fois.

Supposons que nous voulions imprimer les nombres de 1 à 10. Nous pourrions procéder ainsi :

```
10 X=1
20 PRINT X
30 X=2
40 PRINT X
50  .
    .
    etc
```

Il y a toutefois une méthode qui utilise moins d'instructions. Essayons :

```
10 X=1
20 PRINT X
30 X=X+1
40 IF X<11 THEN 20
```

Ce programme contient deux points importants. D'abord la ligne 30 semble être un non sens mathématique. X ne peut pas égaler X+1. Mais pour un ordinateur, cela signifie "que la position de mémoire appelée X prenne l'ancienne valeur de X auquel on ajoute 1". En d'autres termes, "faisons en sorte que X devienne X+1". De quelque manière que nous le disions, l'effet de la ligne 30 est d'ajouter 1 à X. A la ligne 10 nous avons mis X à la valeur 1, si bien que la première fois que nous exécuterons la ligne 30, X prendra comme valeur 2. La prochaine fois que nous exécuterons la ligne 30 la ligne aura pour valeur 3 et ainsi de suite.

Deuxièmement, la ligne 40 contient IF X<11 THEN... Le programme s'arrête quand X est égal à 11 ou est plus grand que 11. Nous souhaitons que le programme imprime les nombres jusqu'à 10 inclusivement et de ce fait nous l'arrêtons quand X a atteint la valeur de X+1, c'est-à-dire 11. Une autre erreur courante dans les programmes d'ordinateur est de faire exécuter une boucle une fois de plus que nécessaire ou une fois de moins que nécessaire.

Nous pouvons utiliser un programme similaire pour imprimer tous les nombres pairs compris entre 1 et 10 inclusivement.

```
10 X=2
20 PRINT X
30 X=X+2
40 IF X>10 THEN STOP ELSE 20
```

Il est important d'examiner la ligne 40 de nouveau. Si nous avions l'instruction X=11 THEN STOP ou quelque chose de ce genre, le programme ne pourrait jamais s'arrêter car X n'est jamais égal à 11 ; il passe de 10 à 12. Pour des raisons de sécurité nous pourrions utiliser le signe >= pour arrêter la boucle.

Nous pouvons utiliser le même genre de programme pour effectuer le "compte à rebours" du lancement d'une fusée.

```
10 X=10
20 PRINTX
30 X=X-1
40 IF X<=0 THEN PRINT "FEU":STOP ELSE 20
```

LIGNES CONTENANT DES INSTRUCTIONS MULTIPLES

Maintenant j'ai fait quelque chose de plus à la ligne 40. Non seulement j'ai été excessivement prudent en utilisant "X<=0" alors que "X=0" aurait suffi, mais j'ai aussi utilisé un ":" et une commande supplémentaire.

Généralement, si la condition suivant le "IF" n'est pas vraie, alors le reste de la ligne est ignorée jusqu'à ce que ELSE soit rencontré. Si la condition suivant le "IF" est vraie, alors le reste de la ligne précédent le ELSE est exécuté.

FOR...NEXT

Les boucles de programme sont si importantes que des instructions spéciales ont été conçues pour cela. Les boucles FOR...NEXT font exactement ce que nous avons fait jusqu'à maintenant mais sans les instructions "IF". Ces boucles ont déjà été mentionnées brièvement au chapitre 5.

```
10 FOR X=1 TO 10
20 PRINT X
30 NEXT
```

Cela imprimera les nombres de 1 à 10. En fait, toutes les lignes comprises entre le FOR et le NEXT seront exécutées 10 fois.

STEP

Nous avons utilisé l'instruction X=X+2 de manière à imprimer par incrément de deux. L'instruction FOR...NEXT peut aussi être utilisée pour faire cela en définissant le pas, l'incrément (STEP).

```
10 FOR X=2 TO 10 STEP 2
20 PRINT X
30 NEXT
```

Nous pouvons faire un "compte à rebours" de cette manière :

```
10 FOR X=10 TO 1 STEP-1
20 PRINT X
30 NEXT X
40 PRINT "FEU"
```

Si vous n'utilisez pas STEP dans l'instruction FOR, alors l'ordinateur fait l'hypothèse que vous voulez dire STEP+1 (valeur par défaut).

Cela peut conduire à des résultats bizarres. La boucle FOR...NEXT est toujours exécutée au moins une fois même si elle ne le devrait pas. Ainsi :

```
10 FOR X=10 TO 1
20 PRINT X
30 NEXT
```

imprimera 10 puis continuera. X vaudra 11 et comme c'est plus grand que 1 l'ordinateur pensera qu'il a fini la boucle.

Le pas n'a pas besoin d'être un nombre entier. Ainsi :

```
10 FOR X=0.014 TO 0.131 STEP 0.017
20 PRINT X
30 NEXT
```

fonctionnera aussi.

Nous pouvons même utiliser les variables dans la boucle. Essayez :

```
10 INPUT"COMBIEN";Y%
20 FOR X=1 TO Y%
30 PRINT X
40 NEXT
```

Comme nous traitons le sujet des boucles, il y a deux choses que vous ne devriez jamais faire.

1. Ne réinitialisez pas continuellement la variable servant d'indice à la boucle, à une valeur constante à l'intérieur de la boucle.

```
10 FOR X=1 TO 30
20 X=5
30 PRINT X
40 NEXT
```

2. Ne sautez pas à l'intérieur d'une boucle.

```
10 GOTO 30
20 FOR X=1 TO 10
30 PRINT X
40 NEXT
```

Je vous laisse comprendre pourquoi il ne faut pas le faire. Et si vous ne comprenez pas essayez les programmes tels quels et ORIC vous le dira.

Il est possible "**d'emboîter**" les boucles les unes dans les autres, ceci n'est pas un mot décrivant une instruction mais réfère à la structure d'un programme (une manière de construire les programmes).

```
10 FOR Y=0 TO 10
20 FOR X=0 TO 10
30 PLOT X,Y,"*"
40 NEXT X
50 NEXT Y
```

La boucle relative à X, les lignes 20, 30 et 40 sont "emboîtées" à l'intérieur des lignes 10 et 50. Aussi pour chacune des valeurs de Y allant de 0 à 10, X varie aussi de 0 à 10. Cela peut être représenté par le diagramme suivant :

```
┌── BOUCLE Y
│
│ ┌ BOUCLE X
│
└── BOUCLE Y
```

Les lignes 40 et 50 peuvent être abrégées comme ceci :

```
40 NEXT X,Y
```

Le programme ci-dessous ne fonctionnera pas :

```
                    10 FOR X=0 TO 100
 ┌──────────────────
 │ Boucle Y  ────────20 FOR Y=0 TO 100
Boucle X     │      30 PLOT X,Y,"*"
 └──────────────────40 NEXT X
              └──────50 NEXT Y
```

car les boucles ne sont pas emboîtées mais se chevauchent.

Nous pouvons avoir plusieurs niveaux d'emboîtement si nous le souhaitons. Par exemple :

```
                                    10 FOR X=1 TO 9
 ┌──────────────────────────────────
 │                                  20 FOR Y=1 TO 9
 │  ┌───────────────────────────────
 │  │                               30 FOR Z=1 TO 9
 │  │  ┌────────────────────────────
Boucle X  Boucle Y  Boucle Z        40 PRINT X,Y,Z;
 │  │  └────────────────────────────50 NEXT Z
 │  └───────────────────────────────60 NEXT Y
 └──────────────────────────────────70 NEXT X
```

A nouveau les lignes 50, 60 et 70 peuvent être réunies en une seule :

```
50 NEXT Z,Y,X
```

REPEAT...UNTIL

Parfois nous ne savons pas exactement combien de fois nous souhaitons exécuter une boucle. Ce que nous souhaitons faire est de répéter une action ou un nombre d'actions jusqu'à ce qu'une condition soit satisfaite. Si ORIC pouvait battre le fer ou reconnaître la chaleur, il pourrait reformuler un vieux proverbe :

REPEAT battre le fer UNTIL NOT Température = chaud

D'une manière plus réaliste nous pouvons utiliser la boucle REPEAT...UNTIL pour détecter qu'un nombre particulier est entré :

```
10 REPEAT
20 INPUT"QUEL EST MON AGE";A
30 UNTIL A=16
40 PRINT"TRES JUSTE,16 ANS"
```

Un usage courant de REPEAT...UNTIL est de permettre d'entrer des données du clavier au terminal jusqu'à ce qu'une "fin de données" soit rentrée. Supposons par exemple que nous souhaitons faire la moyenne d'un nombre inconnu de valeurs. Nous savons qu'aucune de ces valeurs ne sera 99, et ainsi nous pouvons utiliser 99 comme "fin de données" - c'est-à-dire que nous entrerons 99 quand nous aurons rentré toutes les autres valeurs.

Un programme illustre ceci :

```
10 REPEAT
20 T=T+V
30 N=N+1
40 INPUT"VALEUR SUIVANTE";V
50 UNTIL V=99
60 PRINT "LA MOYENNE EST "T/(N-1)
```

(Si vous avez la machine V1.1 vous n'avez pas besoin de l'espace après EST à la ligne 60).

Ce programme fonctionne à condition qu'au moins une valeur soit entrée avant que les données soient terminées par l'entrée de 99.

NOTE - Dans les programmes similaires à celui ci-dessus, je mettrai des espaces à la fin d'une chaîne de caractères précédent un nombre. Ces espaces sont nécessaires sur la machine V1.0. Si vous avez la machine V1.1 vous pourrez les omettre.

Les boucles REPEAT...UNTIL peuvent contenir des boucles FOR...NEXT emboîtées à l'intérieur de celles-ci et vice-versa. Les boucles REPEAT...UNTIL peuvent être emboîtées les unes dans les autres de la même manière que les boucles FOR...NEXT. Par exemple, essayez :

```
10 REPEAT
20 : T=0:N=0
30 : REPEAT
40 :  T=T+V
50 :  N=N+1
60 :  INPUT"VALEUR SUIVANTE";V
70 : UNTIL V=99
80 : PRINT"IA MOYENNE EST "T/(N-1)
90 UNTIL T/(N-1)=20
100 PRINT"C'EST CE QUE NOUS VOULONS"
```

J'ai utilisé les deux points directement après les numéros de lignes 20 à 80. Cela permet de placer des espaces avant l'instruction de telle manière que l'on obtienne un listing avec indentation. Lorsque vous avez des boucles imbriquées, un listing avec indentation rend vos programmes plus faci les à comprendre.

Si vous souhaitez faire une boucle continue, alors REPEAT...UNTIL 0 peut être utilisée.

SOUS-PROGRAMMES

Les boucles dans les programmes sont très utiles quand nous voulons faire la même chose plusieurs fois. Toutefois il y a des cas où nous souhaitons faire un certain nombre de choses plusieurs fois dans un programme mais pas dans une même boucle, mais à des endroits différents.

Supposons par exemple que nous voulions ajouter 25 % de marge au coût d'achat d'un certain nombre d'articles pour obtenir le prix de vente, et imprimer le résultat. Cela pourrait être fait par :

```
100 INPUT "COUT";C
110 D=0.25*C
120 P=C+D
130 PRINT"PRIX DE VENTE="P
140 PRINT"REMISE="D
```

Dans un grand programme nous pourrions avoir besoin de faire cela 10 ou 20 fois. Cela fait pas mal de travail; aussi ORIC nous permet de traiter ce genre de programme comme un sous-programme (en anglais **subroutine**). Nous "appelons" un sous-programme par l'instruction GOSUB suivie du numéro de ligne où le sous-programme commence.

Le programme suivant utilise des sous-programmes et présente plusieurs points d'intérêts. Il fut écrit par les gentils habitants d'un paradis tropical. Bien que la tribu cannibale de l'île voisine considère que ce peuple est primitif, ce programme montre que leurs préférences sont sensées.

Le programme utilise des variables de chaînes de caractères que nous avons vues avec l'instruction INPUT au dernier chapitre. Il utilise aussi des variables entières de telle manière que toutes les réponses soient des nombres entiers. Ceci est dû au fait que les coquillages cassés ne sont pas une monnaie négociable.

```
10 CLS
20 A$="NOIX DE COCO":B$="COQUILLAGES, LA DOUZAINE"
30 GOSUB 200
40 A$="CHEVRES":B$="COQUILLAGES, L'UNITE
50 GOSUB 200
60 A$="MARIS":B$="COQUILLAGES, L'UNITE"
70 GOSUB 200
80 A$="POLITICIENS":B$="COQUILLLAGES, LA GROSSE"
90 GOSUB 200
100 STOP
110 REM ##########################
200 REM SOUS-PROGRAMME
210 PRINT A$ "EN" B$
220 INPUT C%
230 D%=0.25*C%
240 P%=C%+D%
250 PRINT"LE PRIX EST "P%B$
260 PRINT"LA MARGE EST "D%B$
270 PRINT
280 RETURN
```

```
NOIX DE COCO EN COQUILLAGES,LA DOUZAINE
? 40
LE PRIX EST 50 COQUILLAGES,LA DOUZAINE
LA MARGE EST 10 COQUILLAGES,LA DOUZAINE

CHEVRES EN COQUILLAGES,L'UNITE
? 85
LE PRIX EST 106 COQUILLAGES,L'UNITE
LA MARGE EST 21 COQUILLAGES,L'UNITE

MARIS EN COQUILLAGES,LA DOUZAINE
? 32
LE PRIX EST 40 COQUILLAGES,LA DOUZAINE
LA MARGE EST 8 COQUILLAGES,LA DOUZAINE

POLITICIENS EN COQUILLAGES,LA GROSSE
? 15
LE PRIX EST 18 COQUILLAGES,LA GROSSE
LA MARGE EST 3 COQUILLAGES,LA GROSSE

 BREAK IN 100
Ready
```

Notez l'instruction RETURN à la ligne 280. Elle indique à ORIC de retourner du sous-programme à la prochaine instruction qui suit l'appel fait par l'instruction GOSUB. Tous les sous-programmes de ce type doivent finir par une instruction RETURN.

J'ai "appelé" le sous-programme aux lignes 30,50,70 et 90; une fois qu'il a été exécuté, le programme est revenu aux lignes 40,60,80 et 100 respectivement.

Je dois m'assurer que le programme ne "tomberait" pas accidentellement dans le sous-programme sans l'avoir appelé et j'ai empêché cela par l'instruction STOP à la ligne 100. Autrement le programme aurait été en ligne 110 après être revenu en ligne 100, au retour du sous-programme appelé à la ligne 90.

Si l'instruction STOP n'avait pas été là le programme aurait provoqué une erreur quand il aurait atteint l'instruction RETURN à la ligne 280. Essayez-le. Souvenez-vous que vous ne pouvez pas endommager votre ordinateur en faisant des erreurs de programmation.

Il est possible qu'un sous-programme en appelle un autre. Par exemple :

```
10 INPUT"UN NOMBRE ENTRE 1 ET 5";N
20 GOSUB 100
30 PRINT N:STOP
100 REM VERIFIE LE NOMBRE
110 IF N<1 OR N>5, THEN GOSUB 200
120 RETURN
200 REM NOMBRE ERRONE
210 PRINT"NOMBRE NON COMPRIS ENTRE 1 ET 5"
220 PRINT"LE NOMBRE 4 EST PRIS PAR DEFAUT":N=4
230 RETURN
```

Un sous-programme peut même s'appeler lui-même. Cela est appelé la récursion. Par exemple :

```
10 PRINT"SELECTIONNEZ LE SERVICE DEMANDE"
20 PRINT"1.ETAT DE COMPTE"
30 PRINT"2.DEPOT"
40 PRINT"3.RETRAIT"
50 GOSUB 100
60 IF N=1% THEN PRINT"VOUS N'AVEZ PAS LES MOYENS POUR
   DEMANDER UN ETAT DE COMPTE"
70 IF N=2% THEN PRINT"DONNEZ-MOI L'ARGENT"
80 IF N=3% THEN PRINT"VOUS PLAISANTEZ J'ESPERE"
90 STOP
100 REM DEMANDE ET VERIFIE LE NOMBRE
110 INPUT"VOTRE CHOIX";N%
120 IF N%<1 OR N%>3 THEN PRINT"MAUVAISE
    SELECTION":GOSUB 100
130 RETURN
```

ON...GOTO

Il y a une manière très claire de construire un programme qui permet à un utilisateur de faire un choix parmi une liste ou un menu affiché à l'écran. C'est l'instruction ON...GOTO qui est de la forme :

```
ON N GOTO 100,200,300
```

Elle permettra au programme de sauter à la ligne 100 si N=1, et à la ligne 200 si N=2, et à la ligne 300 si N=3. Si N n'est pas un nombre entier, il sera arrondi. Si N est en dehors de l'intervalle spécifié (ici A13) alors le programme exécutera l'instruction suivant directement l'instruction ON...GOTO.

Le programme suivant fait la même chose que le précédent mais utilise l'instruction ON...GOTO.

```
10 PRINT"SELECTIONNEZ LE SERVICE DEMANDE"
20 PRINT"1.ETAT DE COMPTE"
30 PRINT"2.DEPOT"
40 PRINT"3.RETRAIT"
50 INPUT"VOTRE CHOIX";N
60 ON N GOTO 100,110,120
70 PRINT"ENTREE INVALIDE":GOTO 50
100 PRINT"VOUS N'AVEZ PAS LES MOYENS POUR DEMANDER UN
    ETAT DE COMPTE":STOP
110 PRINT"DONNEZ-MOI L'ARGENT":STOP
120 PRINT"VOUS PLAISANTEZ J'ESPERE":STOP
```

ON...GOSUB

ON...GOSUB fonctionne d'une manière très similaire à ON...GOTO, mais est encore plus utile. Plutôt que de sauter simplement à des lignes spécifiées en fonction de la valeur de la variable, il saute à des sous-programmes. Par exemple :

```
10 REPEAT
20 INPUT"QUEL CHIFFRE";N
30 ON N GOSUB 100,200,300,400
40 PRINT
50 UNTIL F=1:END
100 PRINT"UN FOIE":RETURN
200 PRINT"DEUX REINS":RETURN
300 PRINT"TROIS RAISONS":RETURN
400 PRINT"DE BOIRE":F=1:RETURN
```

END à la ligne 50 fait à peu près la même chose que STOP, excepté que après END le programme ne peut pas être relancé avec CONT.

TRON et TROFF

Du fait qu'il y a des décisions dans les programmes ceux-ci ne s'exécutent plus dans l'ordre strict des lignes. Par exemple le dernier programme peut aller à l'une quelconque des lignes 100,200,300,400 avant d'aller à la ligne 40.

Quand vous mettez au point de tels programmes, il est souvent utile de retracer l'ordre exact dans lequel les lignes sont "exécutées". Pour ce faire vous pouvez mettre l'instruction TRON (abrégé pour TRACE ON, positionnement de la trace) dans un programme. Essayez de voir le résultat de l'adjonction de la ligne :

```
5 TRON
```

au dernier programme. A chaque fois qu'une ligne est exécutée le numéro de cette ligne est imprimé sur l'écran entre crochets. Notez que TRON ne peut être utilisé qu'à l'intérieur d'une ligne de programme. Ce n'est pas une commande immédiate.

```
RUN
[ 10][ 20]QUEL CHIFFRE? 1
[ 30][ 100]UN FOIE
[ 100][ 40]
[ 50][ 20]QUEL CHIFFRE? 2
[ 30][ 200]DEUX REINS
[ 200][ 40]
[ 50][ 20]QUEL CHIFFRE? 3
[ 30][ 300]TROIS RAISONS
[ 300][ 40]
[ 50][ 20]QUEL CHIFFRE? 4
[ 30][ 400]DE BOIRE
[ 400][ 400][ 40]
[ 50][ 50]
Ready
```

Si vous souhaitez ne suivre la trace d'exécution que d'une partie de programme, alors l'instruction TROFF supprime l'impression de la trace pour cette partie. Essayez le résultat produit par l'insertion de TROFF à différents endroits dans le programme. La commande NEW supprime le positionnement de la trace en même temps qu'elle efface l'espace mémoire programme.

Chapitre 9

QUELQUES MOTS SUR LES MOTS

CHAINES

Les mots dans l'ORIC sont souvent appelés "Chaînes" car ce sont des chaînes de caractères. Ainsi "HELAS PAUVRE ORIC" est une chaîne de caractères qui a 17 caractères de long. Souvenez-vous qu'un espace est aussi un caractère. En fait tout ce que vous tapez entre les guillemets, que ce soit un nombre, une lettre, un espace, un symbole ou quelque chose d'autre, fait partie de la chaîne de caractères. Pour stocker cette chaîne de caractères dans l'ordinateur nous lui donnons un nom. Ce nom est appelé une **variable alphanumérique** (ou variable de chaînes de caractères). ORIC a un certain nombre de règles assez strictes à propos des variables que nous étudierons plus tard dans ce chapitre. En attendant nous prendrons le chemin le plus facile et nous représenterons des variables de chaînes de caractères par une seule lettre majuscule suivie par le symbole $ de manière à représenter une chaîne de caractères.

Essayez l'exemple suivant :

```
10 A$="CECI EST UNE CHAINE DE CARACTERES"
20 B$="Cela aussi"
30 C$="Des nombres comme 1,2 et même 3*4/6"
40 PRINT A$
50 PRINT B$
60 PRINT C$
```

Vous verrez qu'une chaîne de caractères peut contenir à la fois des caractères minuscules et des caractères majuscules. Toutefois le nom d'une variable alphanumérique doit être composé de lettres majuscules -a$="faux" n'est pas acceptable pour ORIC.

CONCATENATION

Nous pouvons additionner des chaînes de caractères ensemble. Ceci est appelé la concaténation. Par exemple :

```
10 A$="Salut"
20 B$="Je suis ORIC"
30 C$=A$+B$
40 PRINT C$
```

imprimera "SalutJe suis ORIC". Parce que nous n'avons pas ajouté d'espace entre "Salut" et "Je suis" aucun espace n'a été inséré.

Tout caractère créé par CHR$ est lui-même une chaîne de caractères et peut être concaténé à d'autres chaînes de caractères. C'est une caractéristique particulièrement utile si nous voulons mettre un bloc de caractères sur deux lignes à l'écran. Ceci est mieux illustré par un exemple.

```
10 A$=CHR$(27)+"I7K"
20 REM DEUX CARACTERES ALTERNATIFS
30 A$=A$+CHR$(10)+CHR$(8)+CHR$(8)+CHR$(8)
40 REM UNE LIGNE PLUS BAS ET TROIS ESPACES
50 A$=A$+CHR$(27)+"IUZ"
60 REM DEUX CARACTERES ALTERNATIFS DE PLUS
70 CLS :PRINT CHR$(17)'CURSEUR CACHE
80 PRINT:PRINT:PRINT SPC(17)A$
90 GOTO 90
100 REM ARRET DU PROGRAMME PAR CTRL C
```

FRE

Une autre caractéristique intéressante de ce programme est que la variable alphanumérique A$ a été utilisée pour contenir des chaînes de caractères de plus en plus longues, son contenu étant égal à son contenu précédent plus quelques nouveaux caractères. C'est une méthode très pratique pour constituer de longues chaînes de caractères mais cela a un sérieux désavantage. Cette méthode utilise beaucoup d'espace mémoire. C'est parce que toutes les versions successives de A$, et pas seulement la dernière version, sont contenues en mémoire. Cela n'a pas d'importance dans un petit programme mais dans un gros programme cela peut entraîner un manque d'espace mémoire.

L'instruction FRE vous permet de savoir quel espace mémoire reste disponible. Par exemple :

```
PRINT FRE(0)
```

utilisée comme commande immédiate vous dit le nombre d'"octets" restant disponibles en mémoire, et l'instruction comme :

```
100 IF FRE(0)<150 THEN PRINT "PLUS D'ENTREES"
```

peut être utilisée dans un programme d'entrée de données.
Une forme de la commande encore plus utile est :

```
FRE("")
```

Si elle est insérée dans un programme elle effacera de la mémoire toutes les versions des chaînes de caractères exceptées les versions finales.

DECISIONS UTILISANT DES CHAINES DE CARACTERES

Nous pouvons prendre des décisions en utilisant des chaînes de caractères.

```
10 INPUT"ENCORE-O/N";R$
20 IF R$="O" THEN PRINT "C'EST FAIT":GOTO 10 ELSE STOP
```

Pour que les chaînes de caractères soient égales elles doivent contenir exactement le même nombre de caractères dans le même ordre. Les espaces ont de l'importance et les lettres majuscules ne sont pas égales aux lettres minuscules. Si dans le dernier programme vous aviez entré "OUI" au lieu de "O", le programme se serait arrêté car "O" n'est pas égal à "OUI".

COMPARAISON DE CHAINES DE CARACTERES

Nous pouvons comparer des chaînes de caractères en utilisant les symboles plus grand que (>) et moins grand que (<). Ces symboles ne se refèrent pas à la longueur des chaînes de caractères mais au contraire aux valeurs ASCII contenues dans la chaîne de caractères.

Par exemple la valeur ASCII de A est 65 et celle de B est 66.

Ainsi "A" est inférieur à "B".

Si le premier caractère des deux chaînes est identique, le deuxième caractère est comparé et ainsi de suite jusqu'à ce qu'il y ait une différence.

Ainsi "AA" est inférieur à "AB"; "AAB" est plus grand que "AAA".

Le programme suivant vous permet d'entrer et de comparer deux chaînes de caractères quelconques. Utilisez-le jusqu'à ce que vous soyez familiarisé avec cette idée. Notez que la comparaison est faite sur le contenu des variables alphanumériques et non sur les lettres utilisées pour dénommer ces variables alphanumériques.

```
10 REPEAT
20 INPUT"PREMIERE CHAINE";A$
30 INPUT"SECONDE CHAINE";B$
40 IF A$<B$ THEN PRINT A$"<"B$
50 IF A$>B$ THEN PRINT A$">"B$
60 IF A$=B$ THEN PRINT A$"="B$
70 UNTIL B$="STOP":STOP
```

Arrêtez le programme en entrant "STOP" comme seconde chaîne de caractères.

Comme l'ordre alphabétique des caractères A à Z correspond à l'ordre de leur code ASCII, nous pouvons utiliser les comparaisons pour ordonner les chaînes de caractères en ordre alphabétique. Toutefois, faites attention de vous souvenir que les lettres minuscules ont toutes des codes ASCII dont la valeur est supérieure à une quelconque lettre majuscule, c'est ainsi que "a" est plus grand que "Z".

Si vous préférez :

if A$="MAL" et B$="mal" alors A$ est le moindre "mal";
if A$="Bien" et B$="BIEN" alors A$ est le plus grand "Bien".

VAL

VAL change une chaîne de caractères contenant des chiffres en nombres. Par exemple :

```
10 A$="2"
20 PRINT A$, VAL(A$)
```

imprimera 2 2. Pas très inspirant à première vue. En premier, le programme a imprimé une chaîne de caractères, en deuxième il a imprimé un nombre. Vous ne pouvez mélanger chaînes de caractères et nombres; ainsi, vous ne pouvez écrire PRINT 2+(A$)dans la même expression, mais vous pouvez écrire PRINT 2+VAL(A$).

VAL agit également sur les symboles mathématiques tels que +, -, * et /. Ainsi :

```
10 A$="5+14-6*3"
20 PRINT A$"="VAL(A$)
```

imprimera 5+14-6*3=1.

Si VAL est utilisée sur de l'alphabétique, elle fournit comme résultat zéro; même VAL(ORIC)=0.

STR$

STR$ fait le contraire de VAL : elle convertit un nombre en chaînes de caractères.

Ainsi STR$(1234)="1234".

Un des avantages qu'il y a à convertir un nombre en chaîne de caractères est que nous pouvons alors utiliser l'instruction PLOT pour la placer n'importe où sur l'écran. Par exemple :

```
PLOT 17,12,1234
```

produit une erreur, tandis que

```
PLOT 17,12,STR$(1234)
```

fournit le résultat recherché.

Note - Dans la machine V1.0, STR$ utilisé avec des nombres positifs insère un caractère illicite au début de la chaîne. Pour corriger cela, utilisez A$=STR$(...):A$=RIGHT$(A$,LEN(A$)- 1). LEN et RIGHT$ sont expliqués dans la page suivante. Cette faute a été corrigée dans la machine V1.1.

Un autre avantage, probablement le plus grand, de la transformation un nombre en chaîne de caractères est que nous pouvons alors utiliser les instructions puissantes de manipulations de caractères qui sont disponibles dans ORIC. Voyons-les.

LEN

D'abord nous pouvons trouver la longueur d'une chaîne de caractères en utilisant l'instruction LEN.

```
10 INPUT"Quel est votre nom";N$
20 PRINT"Votre nom contient "LEN(N$)"lettres"
```

La ligne 20 utilise LEN pour compter le nombre de lettres dans N$ et imprimer ce nombre.

LEFT$

LEFT$ permet d'extraire un certain nombre de caractères depuis le début de la chaîne. Par exemple :

```
PRINT LEFT$("EXEMPLE BETE",7)
```

imprimera "EXEMPLE".

Souvenez-vous que les espaces sont des caractères et sont comptés, si bien que

```
A$ ="UN MEILLEUR EXEMPLE":PRINT LEFT$(A$,5)
```

imprimera "UN ME".

RIGHT$

RIGHT$ permet d'obtenir un certain nombre de caractères à partir de la fin de la chaîne de caractères. Par exemple :

```
PRINT RIGHT$("UNE GRAND-MERE",4)
```

imprimera "MERE".

MID$

LEFT$ et RIGHT$ sont des cas spéciaux de l'instruction plus générale d'extraction des caractères MID$. La forme générale de cette instruction est :

```
MID$(A$,A,B)
```

Ce qui extraira B caractères de la chaîne A$ en commençant à la position A et en se déplaçant vers la droite. Cela peut paraître compliqué mais quelques exemples éclaireront cela. Par exemple :

```
PRINT MID$("12345",3,2)
A$="UNE CHAINE":PRINT MID$(A$,1,3)
B$="J'AIME ORIC":PRINT
   MID$(B$,1,4)+MID$(B$,7,1)+MID$(B$,9,2)
```

imprimera

```
34
UNE
J'AI RI
```

respectivement.

Si vous ne spécifiez pas un deuxième nombre, MID$ extraira une chaîne qui commence à la position sélectionnée A et qui ira jusqu'à la fin de la chaîne de caractères. Par exemple :

```
PRINT MID$ ("FRED",2)
```

imprimera "RED".

MID$ (A$,1,N) est identique à LEFT$(A$,N)
MID$ (A$,N,1) extrait le nième caractère d'une chaîne.

Nous pouvons utiliser MID$ pour imprimer un mot à l'envers comme dans le programme ci-dessous:

```
10 INPUT "QUEL MOT";W$
20 FOR N=LEN(W$) TO 1 STEP -1
30 PRINT MID$(W$,N,1);
40 NEXT
50 PRINT
60 GOTO 10
```

KEY$

Une autre méthode pour entrer une chaîne de caractères au clavier est d'utiliser KEY$.

KEY$, utilisée en tant que partie d'une instruction (par exemple A$=KEY$) balaye le clavier pour voir si une touche est enfoncée. A la différence de l'instruction INPUT, toutefois, elle ne cause pas l'arrêt du programme jusqu'à ce que vous ayez tapé quelque chose suivi par RETURN. Si une touche est enfoncée KEY$ prend la valeur correspondant au caractère de la touche enfoncée. Si aucune touche n'est enfoncée KEY$ prend comme valeur "la chaîne vide".

Par exemple :

Si la touche P est enfoncée KEY$ est égal à "P".
Si la touche 1 est enfoncée KEY$ est égal à "1".
Si aucune touche n'est enfoncée KEY$ est égal à "" (chaîne de caractères vide).

L'utilisation de KEY$ efface le contenu du tampon du clavier - d'où la forme de l'instruction A$=KEY$ qui stocke l'entrée d'une manière plus permanente dans une variable alphanumérique telle que A$. Si une touche est maintenue enfoncée KEY$ ne la détectera pas de nouveau jusqu'à ce que la touche REPEAT soit utilisée et entraîne alors que le caractère soit entré de nouveau dans le tampon du clavier.

Si deux touches (SHIFT ou CTRL non inclus) sont pressées simultanément au même moment, KEY$ ne détectera pas la seconde touche enfoncée jusqu'à ce que la première soit relâchée.

Pour voir les effets de KEY$ utilisez le programme :

```
10 A$=KEY$
20 PRINT A$
30 GOTO 10
```

GET

GET arrête le programme jusqu'à ce qu'une touche autre que SHIFT ou CTRL soit enfoncée. GET est suivi par un nom de variable, par exemple GETA$, GETB.

```
10 GET A$
20 PRINT A$
30 WAIT 200
40 GOTO 10
```

Comparez ce programme avec le programme précédent.

Vous vous demandez peut-être pourquoi j'ai mis la ligne 30 dans ce programme. Vous comprendrez quand vous essaierez d'arrêter le programme en utilisant CTRL C - pas facile n'est-ce pas ?

Vous devez utiliser CTRL C juste après que vous ayez pressé une autre touche. La ligne 30 vous donne le temps de le faire.

Si vous souhaitez que ORIC soit inactif, utilisez :

```
10 GET B$
20 GOTO 10
```

Vous ne pouvez sortir de ce programme à l'aide de CTRL C car l'instruction GET stocke tout simplement cette entrée et toutes les autres dans une variable alphanumérique B$. Pour vous échapper de cette boucle vous devez utiliser l'interrupteur RESET sous l'ordinateur.

Si maintenant on entrait un nombre, GET peut être suivi par une variable numérique plutôt que par une variable alphanumérique. Cela a pour désavantage qu'une erreur "TYPE MISMATCH" résultera habituellement d'une entrée non numérique.

GET est souvent utilisé pour simplement arrêter le programme jusqu'à ce qu'une touche quelconque soit enfoncée. C'est utile si, par exemple, vous affichez beaucoup de texte sur un écran et demandez à l'utilisateur d'appuyer sur une touche quelconque pour continuer une fois qu'il ou elle l'a lu. Voici un autre exemple qui est très populaire auprès de mes enfants mais pas tellement auprès de ma femme. Il utilise quelques instructions que nous n'avons pas encore vues, mais vous verrez qu'il n'est pas très difficile de comprendre ce qu'elles font.

```
10 GET A$
20 ZAP:WAIT 5:SHOOT:WAIT 10:PING:WAIT 5:EXPLODE
30 GOTO 10
```

ASC

L'instruction ASC donne la valeur ASCII du premier caractère d'une chaîne de caractères à laquelle cette instruction est appliquée.

Par exemple, la valeur ASCII associée à "A" est 65.

```
Ainsi ASC("A")=65
      ASC("AUTRE EXEMPLE")=65
```

ASC utilisé avec la chaîne de caractères nulle ou la chaîne de caractères vide "" (pas d'espace entre les guillemets) donne un message d'erreur ILLEGAL QUANTITY ERROR.

CHR$

CHR$ est l'opposé de ASC. Il convertit le code d'une chaîne de caractères en ce caractère.

Ainsi en prenant l'exemple précédent, PRINT CHR$ (65) fait la même chose que PRINT "A". Nous avons déjà discuté de CHR$ en détail au chapitre 7.

NOM DE VARIABLES

Nous savons déjà qu'un nombre peut être stocké dans une variable numérique et une chaîne de caractères dans une variable alphanumérique. Jusqu'à maintenant nous avons utilisé une seule lettre majuscule suivie de $ si la variable est une variable alphanumérique.

Cela a été fait dans un but de simplicité mais ce n'est pas la meilleure pratique de programmation. Il est plus clair, par exemple, d'utiliser NOM$ comme variable contenant des noms, plutôt que juste N$. La ligne :

```
100 PERSONNES=ADULTES + ENFANTS
```

est plus explicite que :

```
100 P=A+E
```

Nous pouvons utiliser un nombre quelconque de caractères (à l'intérieur des limites imposées par le tampon du clavier) pour les noms à la fois des variables numériques et des variables alphanumériques, à condition que nous observions ces règles.

1. Les variables alphanumériques doivent se terminer par $, les variables entières doivent se terminer par %.

2. Toutes les lettres doivent être des lettres majuscules.

3. Les variables peuvent contenir des nombres et des espaces, mais le premier caractère doit être une lettre.

4. Les variables ne peuvent contenir des caractères autres que les lettres, les nombres et les espaces (par exemple !,*='+) excepté pour le $ et pour % qui suivent les variables alphanumériques et entières respectivement.

5. Les noms des variables ne peuvent contenir à aucun endroit des lettres qui forment une instruction ou une fonction utilisée par l'ordinateur. Par exemple, nous ne pouvons utiliser TOTAL car cela commence par TO (qui est utilisé dans les boucles FOR et NEXT), nous ne pouvons utiliser FRED$ car cela commence par FRE, et nous ne pouvons utiliser PALET car cela contient LET.

Enfreindre l'une quelconque de ces règles causerait une erreur de syntaxe. Il y a toutefois une autre restriction. Elle ne cause pas d'erreur de syntaxe mais peut faire que votre programme donne de mauvais résultats. Seuls les **deux** premiers caractères du nom d'une variable sont utilisés par l'ordinateur.

Ainsi la ligne que nous avons vue précédemment : 100 PERSONNES=ADULTES+ENFANTS est lue par ORIC comme 100 PE=AD+EN.

C'est une bonne pratique de programmation de donner à vos variables des noms entièrement significatifs, mais c'est pour votre propre bénéfice et non pour celui de l'ordinateur.

ADULTES et ADOLESCENTS sont deux choses complètement différentes pour nous, mais si elles sont utilisées comme noms de variables elles sont toutes les deux la même chose pour ORIC : la variable de nom AD.

Par exemple :

```
10 ADOLESCENTS = 100
20 ADULTES = 1000
30 PRINT ADOLESCENTS
```

imprimera 1000, et non 100. En ce qui concerne ORIC, la variable AD a été initialisée à 100 à la ligne 10, changée en 1000 à la ligne 20 et imprimée en ligne 30.

Similairement :

```
10 INPUT"PRENOM";NAME1$
20 INPUT"NOM":NAME2$
30 PRINT"SALUT"NAME1$ AND NAME2$
```

donnera un résultat auquel vous ne vous attendez pas. Essayez-le et vous verrez.

CLEAR

Quand vous commencez un programme en utilisant la commande immédiate RUN, toutes les variables numériques sont initialisées à 0 et toutes les variables alphanumériques sont initialisées à la chaîne vide ("")). Vous pouvez souhaiter faire cela en milieu de programme. Dans ce cas, l'instruction utilisée est CLEAR. Essayez :

```
10 N=6:A$="AB"
20 CLEAR
30 PRINT N+1;A$+"A"
```

avec et sans line 20.

CLEAR initialise aussi tous les éléments d'un tableau (voir chapitre suivant).

Chapitre 10

DES NOMBRES AMUSANTS

LES NOMBRES

Il est temps que nous regardions quelques nombres. Comme beaucoup d'entre vous ne sont pas des mathématiciens, j'essaierai de faire en sorte que cela soit aussi peu douloureux que possible.

SGN

SGN nous permet de connaître le signe d'un nombre. Si le nombre est positif alors SGN vaudra +1; si le nombre est négatif SGN vaudra -1. Si le nombre est nul SGN vaudra 0. Par exemple :

```
10 INPUT"QUEL NOMBRE";N
20 IF SGN(N)=0 THEN PRINT"NOMBRE=0"
30 IF SGN(N)=-1 THEN PRINT"C'EST NEGATIF"
40 IF SGN(N)=1 THEN PRINT"C'EST POSITIF"
```

ABS

La fonction ABS ignore complètement le signe et donnera toujours la valeur absolue (positive) d'un nombre. Ainsi ABS(-5) = 5, ABS(6) = 6, ABS(-23.6) = 23.6, et ainsi de suite.

INT

INT ignore le point décimal et les nombres qui viennent après lui. Par exemple :

```
INT(5.1)=5, INT(5.6)=5, INT(5.0)=5
```

INT arrondit toujours au nombre inférieur. Ainsi, INT(-5.1)=-6 (-6 est inférieur à -5.1).

Si vous souhaitez arrondir au nombre entier le plus proche, ajoutez simplement 0.5 au nombre à arrondir. Par exemple, dans le programme suivant :

```
10 INPUT"QUEL NOMBRE";N
20 N=INT(N+0.5)
30 PRINT N
```

N sera toujours arrondi au nombre entier le plus proche.

RND

RND donne un nombre aléatoire compris entre 0 et 1. Ce nombre peut être égal à 0 mais ne peut pas être égal à 1, bien qu'il puisse l'approcher de très près. Un statisticien vous dira que ces nombres ne sont pas tout à fait au hasard.

Vous devez avoir un nombre entre parenthèses après RND. Ce peut être un nombre quelconque, mais pour des fins pratiques, nous n'avons besoin que de RND(1), RND(0) et RND(-1).

RND(1) donne un nombre au hasard compris entre 0 et 1. Il donne un nombre différent chaque fois qu'il est utilisé normalement. Par exemple à chaque fois que vous faites exécuter ce programme :

```
10 FOR N=1 TO 10
20 PRINT RND(1)
30 NEXT
```

vous obtiendrez une séquence différente de nombres au hasard.

RND(-1) "fixe" le générateur de nombres au hasard. Cela signifie qu'il détermine exactement où, dans la séquence des nombres au hasard, RND(1) commencera. Ajoutez la ligne : 5 X=RND(-1) au programme. La première fois que vous ferez exécuter le programme, vous obtiendrez un ensemble de nombre au hasard comme précédemment. Quand vous exécuterez le programme de nouveau, vous obtiendrez le même ensemble de nombres au hasard.

RND(0) donne un nombre au hasard. Toutefois il donne le même nombre au hasard à chaque fois que la même instruction RND(0) du programme est exécutée. Supprimez la ligne 5 et changez la ligne 20 de telle manière que le programme soit :

```
10 FOR N=1 TO 10
20 PRINT RND(0)
30 NEXT
```

Dans le programme suivant l'ordinateur utilise RND(1) dans un jeu de devinettes. Comme RND(1) ne peut égaler 1, INT(RND(1)*10) donne un nombre compris entre 0 et 9 (car INT arrondi à l'entier inférieur).

```
10 CLS
20 PRINT"DEVINEZ UN NOMBRE COMPRIS ENTRE 1 ET 10"
30 X=1+INT(RND(1)*10)
40 : REPEAT
50 : GET A$
60 : Y=VAL(A$)
70 : IF Y>X THEN PRINT Y"EST SUPERIEUR"
80 : IF Y<X THEN PRINT Y"EST INFERIEUR"
90 : TENTATIVES=TENTATIVES + 1
100 : UNTIL Y=X
110 : PRINT"BIEN "Y"EST LA REPONSE"
120 : PRINT"VOUS AVEZ FAIT "TENTATIVES"TENTATIVES"
```

MATHEMATIQUES

Les sept sections suivantes couvrent des sujets mathématiques un peu plus difficiles. Si cela vous convient vous pouvez les sauter et aller à la section "READ, DATA, RESTORE". Si vous faites cela vous disposerez quand même de cette section pour vous y référer, si vous rencontriez dans d'autres programmes des fonctions qu'elles décrivent.

PUISSANCES

Le signe ^ est utilisé pour "élever à la puissance" un nombre, ou "multiplier un nombre par lui-même "n" fois". Ainsi 2^2 = 2 x 2 = 4, ou 7^4 = 7 x 7 x 7 x 7 = 2401. Nous pouvons aussi avoir 2^(1/2) ou 2^(0.5) = 1.4142136. Bien que nous entrions ceci au clavier sous la forme ↑ , il est affiché sur l'écran comme .

SQR

Un nombre élevé à la puissance 1/2 est aussi connu sous le nom de racine carrée de ce nombre. Les racines carrées peuvent aussi être calculées en utilisant l'expression SQR sur l'ORIC. Ainsi pour imprimer les 10 premiers nombres et leurs racines carrées nous pourrions écrire ceci :

```
10 FOR X=1 TO 10
20 PRINT X,SQR(X)
30 NEXT
```

LOG

LOG signifie le logarithme à base 10 d'un nombre. Cela signifie que si 10^X = Y alors LOG(Y) = X. Si vous êtes allés à l'école avant l'ère de la calculatrice de poche, vous avez probablement appris ce logarithme.

EXP et LN

EXP et LN sont deux fonctions mathématiques opposées. LN est le logarithme "naturel". Si (e)^X = Y alors LN(Y) = X, où e est un nombre approximativement égal à 2.718281828. Si vous comprenez cela vous pourriez vouloir utiliser cette fonction dans les programmes mathématiques. Si vous ne le comprenez pas, ne vous en souciez pas. Cela n'a pas beaucoup d'intérêt excepté dans de tels programmes.

EXP signifie exponentielle et décrit la manière dont un certain nombre d'évènements dans la nature sont contrôlés. Par exemple, le déclin de la radio-activité et l'explosion démographique sont exponentielles. Si vous souhaitez étudier celles-ci, EXP sera utile. Si X=LN(Y) alors Y=EXP(X).

PI

PI est aussi disponible sur l'ORIC. PI ou le signe π vaut 3.1415927 approximativement. Vous pouvez l'utiliser pour calculer les aires et les circonférences de cercle. L'aire d'un cercle est PI x rayon x rayon. La circonférence est 2 x PI x rayon.

```
10 INPUT"RAYON DU CERCLE";RAD
20 PRINT "AIRE"=RAD 2*PI
30 PRINT "CIRCONFERENCE" "2*PI*RAYON
```

SIN, COS, TAN, ATN

Parler de PI et de cercles amène à la trigonométrie. ORIC vous permet de calculer le sinus (SIN), le cosinus (COS) et la tangente (TAN). L'inverse de TAN, l'arc-tangente (ATN) est aussi disponible. Si vous savez ce que je veux dire, je n'ai pas besoin de continuer. Si vous n'êtes pas familier avec ces termes, pas d'importance, ils ne sont pas nécessaires pour de nombreux programmes ; excepté pour SIN qui nous est très utile pour dessiner les courbes. Nous étudierons cela au chapitre 12.

Je dois ajouter pour ceux qui sont familiers avec la trigonométrie que ORIC utilise des angles exprimés en radians non en degrés. Pour convertir des radiants en degrés, il faut multiplier par PI et diviser par 180 ou si vous préférez un degré est approximativement égal à 0.0174533 radian.

DEF FN

Nous pouvons définir nos propres fonctions mathématiques. La procédure consiste, avant le premier appel de la fonction, àécrire une ligne de la forme :

```
DEF FN... (l'espace est optionnel)
```

Cela signifie définir une fonction. Le nom de la fonction suit les mêmes règles que les noms de variables. Vous pourrez utiliser aussi des parenthèses après le nom. Celle-ci devrait contenir la variable (ou certaines des variables) utilisée dans la fonction. Après cela vous mettez un signe = et vous écrivez la fonction que vous désirez. Par exemple :

```
10 DEF FNA(X)=5+X
```

donnera un résultat égal à 5 plus le nombre compris entre parenthèses à chaque fois qu'elle sera appelée.

Pour appeler une fonction vous la nommez tout simplement. Ainsi :

```
20 PRINT FNA(6)
```

imprimera 11.

```
10 DEF FNS(A)=A^2+B
20 B=6
30 PRINT FNS(2)
40 PRINT FNS(3)
50 B=10
60 PRINT
```

imprimera

```
10
15
14
```

Les fonctions sont utilisées pour des opérations arithmétiques qui se répètent souvent. Nous pourrions, par exemple, calculer les aires et les circonférences en utilisant des fonctions.

```
10 DEF FNCIRCONF(RAD)=2*PI*RAD
20 DEF FNAIRE(RAD)=PI*RAD^2
30 : REPEAT
40 :  INPUT"RAYON";RAD
50 :  PRINT"CIRCONFERENCE="FNCIRCONF(RAD)
60 :  PRINT"SURFACE="FNAIRE(RAD)
70 :  PRINT
80 : UNTIL RAD=0
90 END
```

Entrez un rayon nul pour arrêter le programme.

READ, DATA, RESTORE

Ce sont trois instructions très utiles que nous n'avons pas encore vues jusqu'à présent. Voici les instructions READ, DATA, RESTORE.

L'instruction READ lit à partir d'une instruction DATA. Ainsi :

```
10 DATA 10,20,30,40, SALUT
20 READ X,Y,X,A,B$
```

serait équivalent à :

```
10 X = 10
20 Y = 20
30 Z = 30
40 A = 40
50 B$="SALUT"
```

Les instructions data peuvent se trouver n'importe où dans le programme. Si l'ordinateur rencontre une instruction DATA pendant qu'il exécute un programme, il l'ignore tout simplement. Vous n'avez pas besoin de guillemets (") autour d'une chaîne de caractères dans une instruction DATA à moins que vous ne souhaitiez inclure des espaces précédant ou suivant cette chaîne de caractères.

Il peut être utile de regrouper des données telles que le nom et l'âge de telle manière que vous puissiez les voir dans le programme. Regardez le programme suivant :

```
10 DATA Alain DUPONT,23,Albert DUPOND,48
20 DATA Louis DURAND,27,Emile DUBOIS,15
100 FOR K=1 TO 4
110 READ N$,A
120 PRINT "NOM :"N$,"AGE : "A
130 NEXT
```

Si vous tentez de lire plus de données qu'il n'y en a, vous obtiendrez un message d'erreur "OUT OF DATA".

A chaque fois que vous faites une lecture (READ) vous obtenez la donnée suivante (DATA) où qu'elle soit dans le programme.

Il y a des fois où vous voulez recommencer à lire au début des données. Pour ce faire utilisez la commande RESTORE.

Si vous essayez de lire le même ensemble de données deux fois sans utiliser l'instruction RESTORE vous obtiendrez de nouveau un message d'erreur "OUT OF DATA".

TABLEAUX

Une manière très utile de manipuler des choses dans un ordinateur est d'utiliser des tableaux. Un tableau est un ensemble de nombres tous appelés par le même nom de variable, mais distingués entre eux par un nombre entre parenthèses.

DIM

Pour permettre à l'ORIC de reconnaître un tableau nous devons commencer par l'instruction DIM pour le "dimensionner". DIM A(14) réservera 15 emplacements mémoire appelés A(0) à A(14). Si nous ne dimensionnons pas le tableau, ORIC réservera par défaut 11 emplacements (numérotés de 0 à 10).

Nous pouvons alors garnir le tableau par exemple :

```
10 DIM A(14)
20 A(0)=6
30 A(6)=43
40 A(14)=127
```

Il est préférable de garnir un tableau soit avec une instruction INPUT ou DATA. Par exemple :

```
10 DIM A(14)
20 FOR K=0 TO 14
30 READ A(K)
40 NEXT
50 DATA 1,2,3,5,1,2,4,9,15,6,43,8,20,31
```

Notez comme il est facile de se réferer à A(K) comme le kième nombre stocké dans A. Exécutez ce programme et tapez PRINT A(0), PRINT A(1), PRINT A(2), etc... en tant que commandes directes.

Les noms de tableaux suivent les mêmes règles que les noms des variables. Ainsi, nous pourrions stocker 25 prix par exemple dans un tableau dimensionné par DIM PRIX(24).

Nous pouvons aussi avoir des tableaux avec plus d'une dimension. Par exemple :

```
DIM B(2,2)
```

représente un tableau de 9 emplacements un peu comme une grille de morpion.

```
B(0,0)B(0,1)B(0,2)
B(1,0)B(1,1)B(1,2)
B(2,0)B(2,1)B(2,2)
```

Chaque élément peut contenir un nombre.

Vous pouvez penser à des tableaux à deux dimensions comme des matrices (lignes, colonnes). Toutefois, ORIC peut utiliser des tableaux à plus de deux dimensions :

```
DIM BIGARRAY (3,3,3,3,3)
```

Je ne sais pas ce que vous voudriez faire d'un tel "tableau" mais si vous le désirez c'est possible.

Vous n'avez pas à dimensionner les tableaux à plus d'une dimension à condition qu'aucune de ces dimensions ne soit supérieure à 10. Toutefois, il est plus sage de le faire. Si vous n'aviez pas dimensionné le tableau B du dernier exemple, ORIC aurait fait l'hypothèse qu'il s'agissait d'un tableau à deux dimensions, chaque dimension ayant 11 éléments et aurait réservé l'espace pour 121 éléments au lieu des 9 dont vous aviez réellement besoin. Si vous n'aviez pas dimensionné BIGARRAY, ORIC aurait essayé de réserver un espace pour 11x11x11x11x11 éléments et vous n'auriez plus eu de mémoire disponible. Un élément de tableau prend au moins 2 octets de mémoire.

Les tableaux peuvent contenir des nombres ou des chaînes de caractères. Les tableaux numériques peuvent contenir des réels ou des entiers. Les tableaux des nombres entiers ne peuvent contenir que des nombres entiers. Les tableaux de nombres réels peuvent tenir des nombres réels avec des chiffres après le point décimal (1.2, 0.34). Un tableau de nombres entiers a le symbole % derrière le nom du tableau, un tableau de chaîne de caractères a le symbole $ après le nom du tableau.

Les tableaux de chaîne de caractères peuvent contenir des chaînes de caractères exactement de la même manière que les tableaux numériques contiennent des nombres :

```
10 DIM NOM$(15)
20 FOR K=0 TO 15
30 READ NOM$(K)
40 NEXT
50 DATA PIERRE, SIMON, WILLIAM, ANTOINE, LOUIS
60 DATA JEAN, JEAN-PHILIPPE, YANN, ARTHUR
70 DATA HUGUES, JACQUES, FRANCIS, THOMAS, ERIC, PHILIPPE
80 PRINT NOM$(7)"EST LE PLUS BEAU"
90 REM LA LIGNE 80 EST PLUTOT BETE
```

Si nous le souhaitions, nous pourrions mettre des chaînes de caractères dans les tableaux à plusieurs dimensions. Les tableaux de chaînes de caractères à une seule dimension sont toutefois les plus courants.

DES NOMBRES UN PEU DROLES

Cette section ressemble un peu à des mathématiques ennuyeuses. Je vous demanderai toutefois d'y apporter de l'attention.

Si vous savez comment ORIC compte vous trouverez certainement un certain nombre de choses plus faciles à comprendre.

Prenons n'importe quel nombre : disons 5831.

Il peut s'écrire comme :

```
 5x1000
+8x100
+3x10
+1

ou  5x10x10x10
   +8x10x10
   +3x10
   +1
```

En d'autres termes, comme nous nous déplaçons vers la gauche, chaque position numérique est multipliée par une "puissance" de 10 plus élevée, autrement dit par un nombre dix fois plus grand que le nombre précédent sur sa droite.

Il n'y a rien de "magique" à propos du nombre 10. Nous avons grandi et été habitués à travailler avec les puissances de 10, simplement parce que la plupart d'entre nous ont 10 doigts.

Toutefois les ordinateurs digitaux comme ORIC n'ont que deux doigts ou deux choses pour compter avec. Un signal peut être là ou ne pas être là. Un interrupteur peut être ouvert ou fermé. Une condition peut être "vraie" ou "fausse". Toutes ces choses sont représentées par les chiffres 1(vrai) ou 0(faux).

Finalement ORIC voit toute chose sous forme d'une énorme quantité de 1 et de 0. C'est tout ce qu'il peut reconnaître.

Nous comptons en décimal en utilisant 10 chiffres ou symboles numériques (0 à 9) ORIC compte en binaire : utilisant deux symboles pour compter (0 et 1). Comment peut-il compter en ayant seulement deux symboles ?

Eh bien, 0(décimal) = 0 (binaire)
1(décimal) = 1 (binaire).

Mais que fait ORIC quand il atteint 2 ? Il n'a plus de symboles disponibles, aussi il doit faire comme nous faisons lorsque nous atteignons 10 et que nous n'avons plus de symboles disponibles. Il se déplace d'un caractère sur la gauche. Ainsi :

2(décimal) = 10(binaire)
3(décimal) = 11(binaire)

ORIC manque de symboles à nouveau, aussi il doit encore se déplacer d'un espace sur la gauche.

4 (décimal) = 100 (binaire)
5 (décimal) = 101 (binaire)
6 (décimal) = 110 (binaire)
7 (décimal) = 111 (binaire)

Ainsi 111 (binaire) est :

```
 1x2x2
+1x2
+1
```

qui est égal à 7 (décimal).

J'espère que vous voyez la structure qui émerge.

1111 (binaire) est :

```
 1x2x2x2
+1x2x2
+1x2
+1
```

qui est égal à 15 (décimal).

Chaque déplacement sur la gauche d'une colonne correspond à une puissance de 2 (au lieu d'une puissance de 10) plus grande que celle sur sa droite. Elle progresse ainsi 1,2,4,8,16,32 et ainsi de suite.

Un seul chiffre binaire (1 ou 0) est appelé un bit en abrégé. Chacun des emplacements mémoire de l'ORIC appelés encore octets, contient 8 bits.

Ainsi le nombre maximum qu'un emplacement mémoire peut contenir est :

11111111 (binaire).

Ceci est égal à :

1x1
+1x2
+1x4
+1x8
+1x16
+1x32
+1x64
+1x128 (colonne d'extrême gauche)

c'est-à-dire 255 (décimal).

Ainsi un emplacement mémoire peut tenir, disons le nombre binaire

10100111.

Ceci est égal à :

1+(1x2)+(1x4)+(0x8)+(0x16)+(1x32)+0x64)+(1x128)
= 1+2+4+32+128 = 167 (décimal).

Il est malcommode pour nous d'écrire des nombres tels que 11011011; aussi nous les représentons avec ce qui est appelé les nombres hexadécimaux. Les nombres hexadécimaux sont simplement une manière abrégée d'écrire les nombres binaires. Cela n'est qu'une facilité pour nous - ORIC n'utilise que le binaire.

Pour obtenir un nombre hexadécimal nous groupons le nombre exprimé en binaire en groupe de 4 bits. Ainsi un simple octet, qui comporte 8 bits, est regroupé en deux ensembles de 4 bits.

Par exemple : 1011 0101

Chacun de ces groupes sera représenté par un caractère séparé - ce qui est facile pour les valeurs jusqu'à 9 :

0000=0
0001=1
.....
1001=9

Pour les nombres binaires de 1010 à 1111 nous utilisons les symboles A à F respectivement.

Ainsi 11 (décimal) = 1011 (binaire) = B (hexadécimal)
16 (décimal) = 0001 0000 (binaire) = 10 (hexadécimal).

ORIC peut travailler avec des nombres hexadécimaux jusqu'à FFFF. Pour indiquer qu'un nombre est hexadécimal, mettez le symbole # devant celui-ci.

Ainsi PRINT#FFFF donnerait 65535
PRINT#1A donnerait 26.

Pour convertir du décimal en hexadécimal, utilisez HEX$. Par exemple PRINT HEX$(26) donne #1A. Comme ni # ni A ne peuvent être imprimés comme des nombres, HEX$ génère une chaîne de caractères. HEX$(0) génère la chaîne "#" dans la machine V1.0 et la chaîne "#0" dans la machine V1.1.

AND, OR

Nous avons vu AND et OR dans le chapitre 8. Ils nous ont permis de prendre des décisions fondées sur deux conditions ou plus. AND et OR sont aussi des fonctions logiques.
Par exemple :

1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0
1 AND 1 = 1

Notez que 1 AND 1 (1 et 1) n'est pas la même chose que 1+1 ! Aussi :

1 OR 1 = 1
0 OR 1 = 1
1 OR 0 = 1
0 OR 0 = 0

AND et OR opèrent sur un bit à la fois. Ainsi :

1011 AND 1101 = 1001 (tout en binaire).

Si vous ne comprenez pas cela, ne vous en souciez pas. Vous pouvez écrire tous vos programmes sans les opérateurs logiques. Mais ils vous donnent une arme de plus dans votre armurerie si vous décidez de les employer.

Par exemple :

```
10 INPUT"NOMBRE ENTRE 0 ET 7";N%
20 N=7 AND N%
```

fera en sorte que N soit un nombre entier compris entre 1 et 7 quelque soit le nombre rentré.

```
10 FOR N=1 TO 20
20 PRINT 10+5*(N AND 1)
30 NEXT
```

imprimera 10 quand N est pair et 15 quand N est impair.

Chapitre 11

LE SON DE L'ORDINATEUR

ORIc a quatre commandes de son prédéfinies, qui correspondent à des bruits couramment utilisés dans les jeux :

ZAP (bruit de rayon laser)
PING (choc)
SHOOT (tir)
EXPLODE (explosion)

Une des bonnes caractéristiques du son d'ORIC est que vous avez juste besoin de l'instruction pour démarrer le son. Pendant que le son est généré le programme peut faire quelque chose d'autre. On peut donc avoir une balle traversant l'écran pendant que SHOOT fait le son.

Si vous voulez une série de sons prédéfinis, vous devez mettre une pause entre chacun afin qu'il ait le temps de finir.

Ainsi :

```
10 FOR N = 1 TO 10
20 PING
30 NEXT
```

ne produira qu'un choc tandis que :

```
10 FOR N = 1 TO 10
20 PING:WAIT 10
30 NEXT
```

produira 10 chocs.

PLAY

Les sons prédéfinis, aussi utiles soient-ils, ne représentent qu'une petite partie du répertoire sonore de l'ORIC. ORIC a trois canaux pour le son, ce qui signifie qu'il peut jouer 3 sons différents à la fois. Chacun de ces 3 canaux peut générer soit des tons purs (musique), soit des tons qui peuvent être mélangés avec du bruit pour produire des effets spéciaux.

Avant d'expérimenter les capacités sonores de l'ORIC, assurez-vous que le cliquetis du clavier est enlevé. CTRL F contrôle le cliquetis du clavier. Si vous obtenez un son continu, alors CTRL F suivi par n'importe quelle touche (RETURN est préférable) le supprimera.

L'instruction PLAY ouvre et ferme les 3 canaux et détermine quel est le canal qui peut être combiné avec le générateur de bruit.

PLAY ne produit pas de son. Il ouvre les canaux qui peuvent être utilisés par les instructions SOUND et MUSIC. L'instruction PLAY le plus couramment utilisée est PLAY 0,0,0,0. Elle ferme tous les canaux.

Quand la machine est mise sous tension le canal 1 est ouvert. Le mélange avec le bruit n'est pas permis.

Le format de l'instruction PLAY est

PLAY TE,NE,EM,EP

où TE,NE,EM,EP sont tous des nombres.

TE (abrégé de Tone Enable) spécifie l'ouverture des canaux et est compris entre 0 et 7.

La table ci-dessous montre les valeurs de TE qui contrôlent les canaux.

TE	CANAL 3	CANAL 2	CANAL 1
0	FERME	FERME	FERME
1	FERME	FERME	OUVERT
2	FERME	OUVERT	FERME
3	FERME	OUVERT	OUVERT
4	OUVERT	FERME	FERME
5	OUVERT	FERME	OUVERT
6	OUVERT	OUVERT	FERME
7	OUVERT	OUVERT	OUVERT

Ainsi par exemple lorsque TE=3, les canaux 1 et 2 sont ouverts et quand TE=7 les 3 canaux sont ouverts.

Comparez les statuts des canaux avec les nombres binaires (étudiés au chapitre précédent). Voyez-vous un rapport ? Essayez en remplaçant "fermé" par 0 et "ouvert" par 1.

NE (abrégé de Noise Enable) sélectionne les canaux sur lesquels on peut mélanger bruit et musique.

NE se comporte comme TE, c'est-à-dire que si NE=3, le bruit peut être mélangé avec les canaux 1 et 2.

Nous étudierons EM et EP bientôt. En attendant laissez les à zéro.

SOUND

Si un canal est ouvert, l'instruction SOUND fait qu'un son est émis par ce canal. L'instruction est de la forme :

SOUND C,P,V

où C,P,V sont des nombres représentant le canal, la période et le volume.

C sélectionne le canal sur lequel l'instruction opère.

Quand C vaut 1 le canal 1 émet de la musique.
Quand C vaut 2 le canal 2 émet de la musique.
Quand C vaut 3 le canal 3 émet de la musique.

Quand C vaut 4 le canal 1 émet de la musique et du bruit.
Quand C vaut 5 le canal 2 émet de la musique et du bruit.
Quand C vaut 6 le canal 3 émet de la musique et du bruit.

V sélectionne le volume et varie de 1 à 15. Quand V vaut 1 le son est faible; une valeur de 3 ou 4 convient à l'expérimentation; une valeur de 6 ou 7 est suffisamment élevée pour la plupart des besoins.

P sélectionne la tonalité de la note. P veut dire période. Cela signifie que plus P est petit, plus le son généré est aigu. Une valeur de P de 238 se situe au milieu du clavier d'un piano. La note la plus basse est générée quand P = 4096. La note la plus aigüe que je puisse entendre est générée par P=4.

Le programme suivant vous permet d'essayer différentes valeurs de P. En appuyant sur n'importe quelle touche, cela arrête le son et vous permet d'entrer une autre valeur pour la période. Une valeur nulle arrête le programme.

```
10 REPEAT
20 INPUT"PERIODE";P
30 IF P=0 THEN STOP
40 PLAY 1,0,0,0
50 SOUND 1,P,4
60 GET A$
70 PLAY 0,0,0,0
80 UNTIL 0
```

Pour essayer l'effet produit par le mélange avec le bruit, changez les lignes 40 et 50 comme suit :

```
40  PLAY 1,1,0,0
50 SOUND 4,P,4
```

Ce programme peut être modifié de telle manière que 3 canaux soient ouverts, avec ou sans mélange de bruit, et que 3 valeurs de période (une pour chaque canal) soient utilisées. Essayez cela vous-même.

ENVELOPPE

Vous vous souvenez que j'ai promis d'expliquer les paramètres EM et EP de l'instruction PLAY. Ceux-ci sont utilisés quand au lieu d'un son à volume constant, nous voulons un son qui varie en volume à différents moments.

Par exemple, nous voulons un son qui croisse graduellement en volume et qui décroisse rapidement (montée lente, descente rapide). Ou bien nous voulons un son qui commence à un volume maximum et cesse graduellement (montée rapide, descente lente).

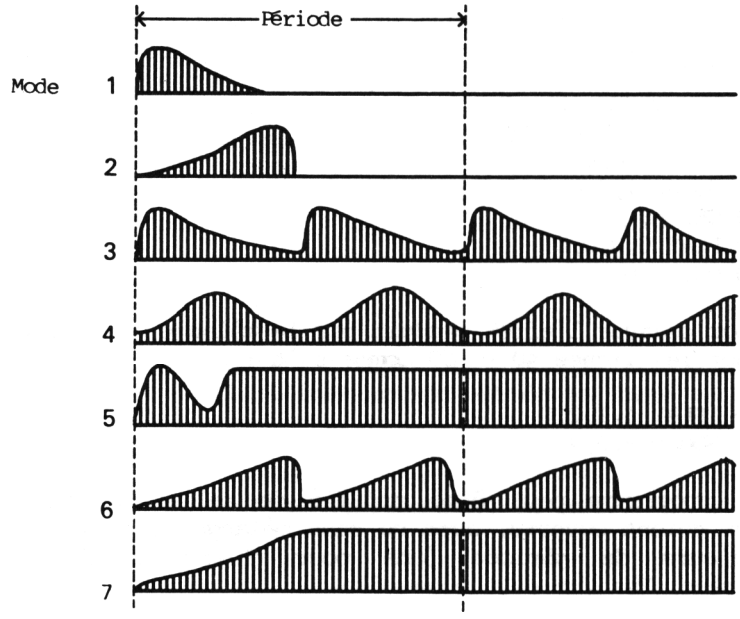
EM est utilisé pour sélectionner l'une des sept "enveloppes" avec des montées et des descentes différentes.

EP est la dure de l'enveloppe mesurée en millième de secondes. Ne confondez pas cela avec WAIT dont la durée est mesurée en centième de seconde.

Le diagramme ci-dessous montre les sept enveloppes disponibles. La longueur de la période est aussi montrée.

Les modes 1 et 2 stoppent le son à mi-durée.

Les modes 3 à 7 sont continus.



Si vous spécifiez une enveloppe dans l'instruction PLAY alors tous les sons émis après cela seront contrôlés par cette enveloppe. Vous pouvez choisir les volumes de vos sons comme précédemment mais si vous le souhaitez vous pouvez spécifier un volume de valeur 0. Ceci permet à l'instruction PLAY de contrôler les volumes et rend "l'enveloppe" plus efficace.

Vous pouvez utiliser le programme suivant pour sélectionner les modes d'enveloppes, les périodes des sons, le mélange ou non avec des bruits. C'est un programme très utile pour essayer des effets sonores. Vous pourriez également faire en sorte de pouvoir entrer la période de "l'enveloppe" au clavier. La frappe d'une touche quelconque arrête le son et permet un autre choix. Le choix d'un mode d'enveloppe nul arrête le programme.

```
10 REPEAT
20 INPUT "MODE DE L'ENVELOPPE";E
30 INPUT "PERIODE DU SON";P
40 INPUT "VOULEZ-VOUS DU BRUIT (O/N)",R$
50 IF R$="O" THEN C=4:N1 ELSE C=1:N=0
60 PLAY 1,N,E,500
70 SOUND C,P,0
80 GET A$
90 PLAY 0,0,0,0
100 UNTIL E=0
```

L'instruction SOUND est utile pour les bruits et les effets spéciaux.

```
10 REM TEUF-TEUF
20 REPEAT
30 PLAY 1,1,2,50
40 SOUND 4,10,0:WAIT 15
50 UNTIL KEY$<>"":REM CHAINE VIDE
60 PLAY 0,0,0,0:REM APPUYEZ SUR N'IMPORTE QUELLE TOUCHE
   POUR FINIR
```

```
10 REM COUREUR
20 REPEAT
30 PLAY 1,1,1,100
40 IF PR=8 THEN PR=14 ELSE PR=8
50 SOUND 4,PR,0:WAIT 20
60 UNTIL KEY$<>"":REM CHAINE VIDE
70 PLAY 0,0,0:REM APPUYEZ SUR N'IMPORTE QUELLE TOUCHE POUR

   FINIR.
```

```
10 REM TAMBOURS
20 REPEAT
30 : FOR N=1 TO 8
40 : PLAY 1,1,1,35
50 : SOUND 4,35,0
60 : WAIT 5
70 : NEXT
80 WAIT 10
90 : FOR N=1 TO 2
100 : PLAY 1,1,1,300
110 : SOUND 4,300*N,0
120 : WAIT 60
130 : NEXT
140 UNTIL KEY$<>"":REM EMPTY STRING
150 PLAY 0,0,0,0 PRESS ANY KEY TO END
```

MUSIC

L'instruction SOUND est utile pour les effets sonores comme ceux que nous avons juste entendus. Elle peut être utilisée pour jouer des harmonies, mais ORIC a une autre instruction, MUSIC, qui est fournie à cet effet.

L'instruction MUSIC prend la forme :

MUSIC C,OC,N,V

où C,OC,N et V sont des nombres. C est le numéro de canal sélectionné, et est un nombre entier compris entre 1 et 6. Cela fonctionne de la même manière que le canal sélectionné par l'instruction SOUND.

OC signifie octave et est un nombre entier compris entre 0 et 6.

N signifie note et est un nombre entier compris entre 1 et 12.

V signifie volume et est un nombre entier compris entre 0 et 15. Cela fonctionne de la même manière que la sélection du volume dans l'instruction SOUND.

Pour que l'instruction MUSIC ait un effet quelconque, les canaux en usage doivent être ouverts. Ceci est fait par l'instruction PLAY. La longueur de la note peut être déterminée par une instruction WAIT. La note peut-être arrêtée en utilisant l'instruction PLAY pour fermer le canal en cause. PLAY 0,0,0,0 supprime tout son.

Avant que nous regardions MUSIC avec plus de détail, essayons ce programme. Il permet d'utiliser ORIC comme un orgue. Mais assurez-vous que le cliquetis du clavier et capitals lock soient enlevés avant d'exécuter le programme. En appuyant simultanément sur la touche SHIFT et sur les touches alphabétiques, vous obtiendrez des notes plus graves.

```
10 REPEAT
20 GET A$:A=ASC(A$)
30 A=ABS(A-32):IF A>83 THEN A=83
40 OCT%=A/12
50 NT=A-12*OCT%+1
60 PLAY 1,0,0,0
70 MUSIC 1,OCT%,NT,5
80 WAIT 50
90 PLAY 0,0,0,0
100 UNTIL A=0'ESPACE ARRETE LE PROGRAMME
```

Ceci est l'essentiel d'un programme qui vous permet d'entendre toutes les notes. Une fois que vous êtes content de l'instruction MUSIC vous pouvez améliorer ce programme de telle manière que les notes soient jouées dans l'ordre dans lequel les touches sont placées sur le clavier, de telle manière que vous puissiez sélectionner le volume et la longueur de la note et aussi que les trois canaux puissent être sélectionnés en même temps.

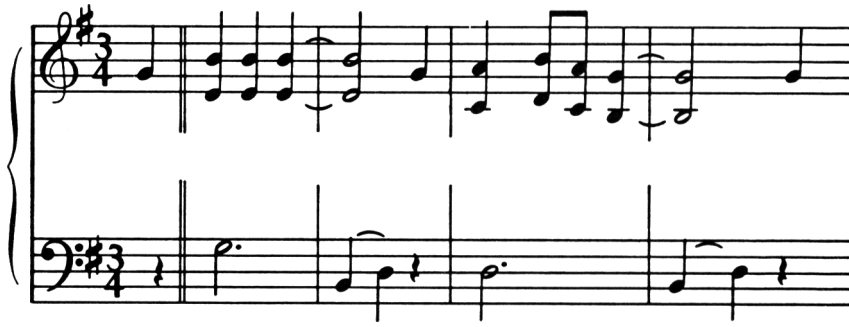
Pour obtenir un effet de "trémolo", remplacez les lignes 60 à 70 par :

```
60 PLAY 1,0,4,100
70 MUSIC 1,OCT%,NT,0
```

THEORIE MUSICALE

Si vous voulez jouer des airs sur ORIC vous avez besoin de connaître un peu de théorie musicale. Si vous connaissez tout de la musique, vous pouvez parcourir cette section très rapidement.

Même un air simple peut sembler décourageant lorsqu'il est sur une feuille de musique. Pour commencer, il y a d'habitude deux ensembles de cinq lignes jointes ensemble.



Vous verrez que l'ensemble des lignes en haut commence par le symbole 𝄞 tandis que l'ensemble des lignes du bas commence par le symbole 𝄢.

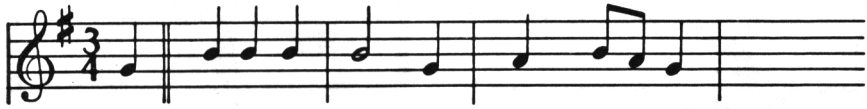
𝄞 est appelé la "clef de sol"
𝄢 est appelé la "clef de fa".

HARMONIE

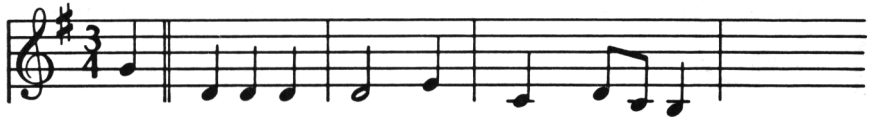
Comme vous le savez probablement les symboles de forme ♩ représentent une note musicale. Toutefois, vous verrez que la plupart des notes dans cet air sont de la forme ♩ .

De nombreuses feuilles de musique que vous rencontrerez ont cette sorte de notes multiples. Ceci est dû au fait que les airs sont écrits pour qu'ils puissent être chantés en harmonie. C'est-à-dire que deux ou plusieurs personnes chantent et que chacun chante une note différente. Vous avez probablement déjà entendu des groupes de chant harmonique. Les notes multiples sont un moyen abrégé de décrire cette musique. Vous pourriez décrire cet air comme :

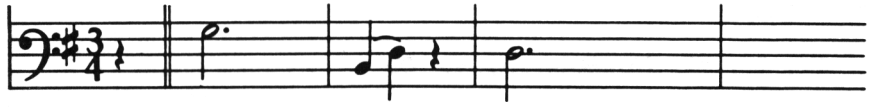
Voix 1



Voix 2



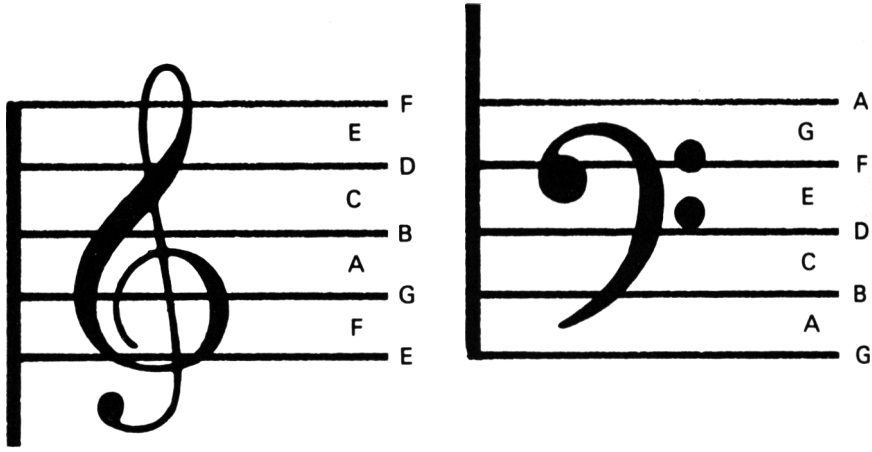
Voix 3



Cet air a trois "voix". ORIC a trois canaux pour le jouer.

HAUTEUR DU SON

Les 5 lignes de la feuille sur lesquelles la musique est écrite est appelée la **portée.** A nouveau vous pouvez savoir que les notes écrites près du haut de la portée sont d'une tonalité plus élevée que celles décrites près du bas de la portée. Pour rendre les choses plus simples les notes ont des noms qui déterminent leur hauteur. Ces noms consistent en lettres de A à G. Donc la position d'une note sur les lignes peut être représentée par une lettre.



Vous avez probablement entendu parler de l'échelle musicale :

DO, RE, MI, FA, SOL, LA, SI, DO

Les 8 notes de l'échelle sont appelées une "octave". L'échelle de "C majeur" est une octave qui commence et qui finit avec C. C'est-à-dire :

C=DO, D=RE, E=MI, F=FA, G=SOL, A=LA, B=SI, C=DO.

De la même manière, une échelle de A commence et finit avec la note A.

Il y a toutefois une complication. Toutes les augmentations de fréquence, (changement de hauteur) ne sont pas les mêmes. Par exemple le changement de hauteur entre les notes B et C représente seulement la moitié du changement de hauteur entre C et D.

Pour permettre cela, une gamme ou octave est divisée en douze "demi-tons". Ce sont :

A
A# ou B♭
B
C
C# ou D♭
D
D# ou E♭
E
F
F# ou G♭
G
G# ou A♭

Le symbole # (dièse) signifie "aigu" et ♭ (bémol) signifie "grave". Ainsi G# est d'un demi-ton plus haut que G et A♭ est un demi-ton plus bas que A. G# donne la même note que A♭.

Je n'explorerai pas les gammes en détail, il suffit de dire que la gamme qui semble bonne pour la plupart d'entre nous est la gamme majeure. Celle-ci monte par demi-ton de la manière suivante :

Do (2 augmentations), Ré (2 augmentations),
Mi (1 augmentation), Fa (2 augmentations),
Sol (2 augmentations), La (2 augmentations),
Si (1 augmentation), Do (2 augmentations).

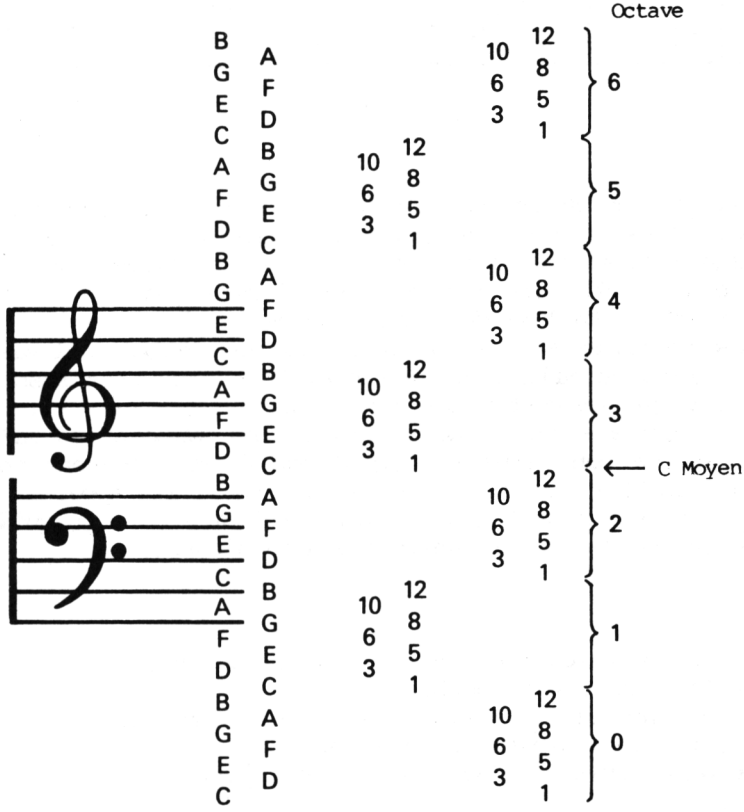
Si bien que la gamme de C majeur est C,D,E,F,G,A,B,C.
Et la gamme de G majeur est G,A,B,C,D,E,F#,G.
Essayez d'autres gammes par vous-même. Vous verrez que seule la gamme de C majeur n'a pas de dièze ni de bémol.

Dans l'air que nous avons étudié, précédemment, vous avez vu que la ligne représentant F avait le symbole # dièze (aigu). Cela signifie que les notes écrites sur ces lignes doivent être jouées comme un F# plutôt que comme un F. Vous pouvez être intéressé par le fait que l'air est écrit avec la clé de G majeur, mais cette connaissance n'est pas essentielle pour convertir de la musique en programme d'ordinateur.

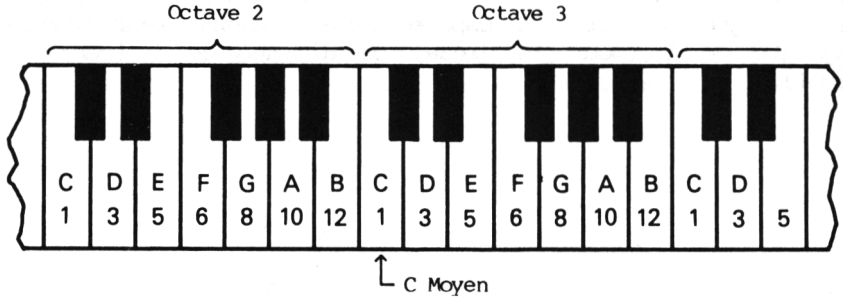
ORIC vous donne 7 octaves complets numérotés de 0 (la plus basse) à 6 (la plus haute). Chaque octave commence à la note C (la plus basse) et finit à la note B (la plus haute). Chaque octave, comme nous le savons, contient douze demi-tons. Aussi le troisième paramètre (N) de l'instruction MUSIC est un nombre de 1 à 12, chaque nombre correspondant à une note.

NOTE	PARAMETRE N
B (la plus haute)	12
A#	11
A	10
G#	9
G	8
F#	7
F	6
E	5
D#	4
D	3
C#	2
C (la plus basse)	1

Le diagramme suivant donne les nombres de l'ORIC correspondant aux positions au-dessus et en-dessous des portées. Quand un dièze ou un bémol est écrit sur la ligne ou sur l'espace au début de la ligne, alors toutes les notes sur cette ligne ou sur cet espace doivent avoir leur nombre augmenté de 1 (pour le dièze) ou diminué de 1 (pour le bémol).



Le C moyen est appelé ainsi parce c'est la note C près du milieu du clavier d'un piano.



S'il y a un dièze ou un bémol affectant une note sur la portée, alors la note correspondant dans les autres octaves sera aussi affectée. Par exemple, si l'air montre que F est positionné à F#, cela signifie que F sera joué comme F# et aura la note 7 au lieu de 6 dans toutes les 7 octaves.

Dans certains airs vous trouverez des symboles dièze et bémol écrits au milieu de la ligne. Vous trouverez aussi le symbole ♮ qui convertit une note à sa valeur "naturelle" si elle est normalement aigue ou grave. Les symboles au milieu de la portée sont appelés "accidentels" et affectent toute note sur la ligne ou sur l'espace sur lequel ils apparaissent jusqu'à la prochaine ligne verticale (c'est-à-dire jusqu'à la fin de la "barre" sur laquelle ils apparaissent).

DUREE

Nous allons regarder la hauteur de la note. Maintenant nous allons étudier sa durée.

Il est normal de définir une note d'un "temps simple" comme ♩ (ceci est appelé un crochet).

Une note durant un "demi-temps" est ♪ (ceci est appelé une croche).

Une note d'un "quart de temps" est ♬ (ceci est appelé une demi-croche).

Parfois, les queues des notes sont jointes ensemble, par exemple :

♫ représente deux notes d'un "demi-temps".
♫ représente deux notes d'un "quart de temps".

Une note durant 2 temps est représentée par 𝅗𝅥 .
Une note durant 4 temps est représentée par 𝅝 .

Un point après la note réduit la note de moitié. Par exemple :

♩. est une note d'un "temps et demi".
𝅗𝅥. est une note de "trois temps".

Parfois vous verrez les notes reliées entre elles par une ligne courbe. Si les notes sont de la même hauteur et contingues les unes aux autres, elles doivent être traitées comme une simple note. Si elles sont de différentes hauteurs, ou s'il y a des notes entre elles, elles doivent être traitées comme deux notes différentes.

Finalement nous en arrivons au "silence". Comme le nom l'indique il y a des pauses dans la musique. Il existe deux silences :

𝄽 - silence pour un temps simple
𝄾 - silence pour un demi-temps.

MESURE

Le chiffre après la clé au début de la musique donne la "mesure" ou le nombre de temps constituant une barre. Habituellement c'est 2/4, 3/4, 6/8 ou 4/4 bien que d'autres valeurs soient possibles. 4/4 est parfois dénoté par C ou ¢. Pour des raisons de simplicité prenez 3/4 comme étant identique à 6/8. Il y a des différences, mais nous n'allons pas plonger trop profondément dans la théorie musicale.

Le nombre de temps dans une barre doit être égal au nombre du haut. Ainsi :

une mesure 2/4 a deux temps par barre;
une mesure 3/4 a trois temps par barre;
une mesure 4/4 a quatre temps par barre.

Toutes les barres complètes ont le même nombre de temps. Il est utile de vous assurer que vous n'avez pas oublié une note ou utilisé une note de longueur incorrecte quand vous programmez un air.

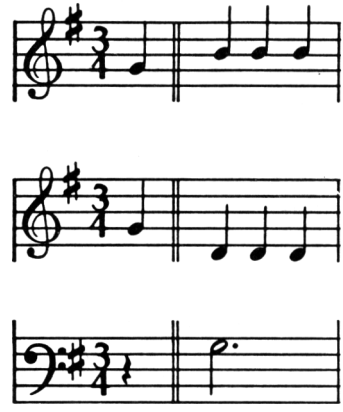
METHODE DE PROGRAMMATION

La première chose que je fais lorsque je transforme une feuille de musique en programme d'ordinateur c'est de trouver la note la plus courte de cette musique. Cela peut être une croche ou une double croche. J'initialise alors une variable TIME égale à la durée exprimée en centièmes de seconde que je souhaite donner à la note la plus courte. Toutes les autres notes dureront un multiple de cette valeur.

Par exemple si la note la plus courte est une demi-croche et que je mets TIME égale à 10, alors un temps complet prendra 4x10 = 40 centièmes ou 0.4 seconde. Si l'air était sur une mesure 3/4 une barre se jouerait en 1.2 seconde.

L'avantage de cette méthode est qu'en changeant la valeur de la variable de TIME qui est fixée au début du programme, je peux accélérer l'air ou le ralentir jusqu'à ce que j'obtienne la vitesse souhaitée.

Regardons l'air avec lequel nous avons commencé. Nous allons le prendre depuis le début jusqu'à la fin de la première barre. Une fois que vous aurez vu comment transformer la notation musicale en instructions de programme, vous utiliserez la même technique pour un air complet.

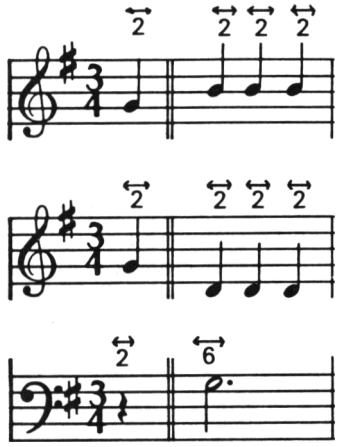


Consacrons les canaux 1 et 2 aux deux voix en clé de sol et le canal 3 à la voix en clé de fa. La note la plus courte dans cet exemple est une croche. Nous initialiserons TIME à 17, ce qui donnera à peu près une durée d'une seconde pour une barre.

La première ligne est :

```
10 TIME=17
```

Partagez la musique en période toutes égales à la note la plus courte.



Dans la première de ces périodes, les voix 1 et 2 jouent toutes deux la note G au-dessus du C moyen qui est l'octave 3 note 8. La voix 3 est silencieuse.

Aussi les prochaines instructions ouvrent les canaux 1 et 2 et jouent les notes choisies.

```
20 PLAY 3,0,0,0
30 MUSIC 1,3,8,5
40 MUSIC 2,3,8,5
```

Ceci doit rester identique pendant deux périodes.

La ligne suivante est :

```
50 WAIT TIME*2
```

Si la note suivante est la même que la précédente mais que nous voulions les entendre comme deux notes séparées, nous avons donc à fermer les canaux appropriés et à les rouvrir de nouveau pendant un court moment. Quand la note suivante est différente nous n'avons pas à faire ceci. Ainsi, dans la période suivante nous n'avons pas à rouvrir les canaux 1 et 2. Par contre, nous devons ouvrir le canal 3. Aussi les lignes suivantes seront :

```
60 PLAY 7,0,0,0
70 MUSIC 1,3,12,5
80 MUSIC 2,3,3,5
90 MUSIC 3,2,8,5
100 WAIT TIME*2
```

Ensuite nous devons fermer les canaux 1 et 2 pour un court moment et les rouvrir ensuite pour obtenir des notes distinctes. Le canal 3 reste ouvert. Ainsi :

```
110 PLAY 4,0,0,0:WAIT 2
120 PLAY 7,0,0,0
130 MUSIC 1,3,12,5
140 MUSIC 1,3,3,5
150 MUSIC 3,2,8,5
160 WAIT TIME*2-2
```

Dans la ligne 160 nous ajustons la longueur de la note pour accommoder le petit délai occasionné par la ligne 110.
Les deux périodes suivantes sont les mêmes. Aussi :

```
170 PLAY 4,0,0,0:WAIT 2
180 PLAY 7,0,0,0
190 MUSIC 1,3,12,5
200 MUSIC 2,3,3,5
210 MUSIC 3,2,8,5
220 WAIT TIME*2-2
```

Finalement, nous avons à arrêter tous les sons :

```
230 PLAY 0,0,0,0
```

Vous pourriez ainsi parcourir un air en entier. Mais ce serait plutôt laborieux. Regardons si nous ne pouvons pas éviter de répéter toutes ces instructions MUSIC et PLAY.

Si le canal est fermé nous pouvons donner une instruction MUSIC pour jouer dessus. Cette instruction n'a pas d'effet. Cela signifie que nous pouvons avoir trois instructions MUSIC à chaque étape bien que tous les canaux ne soient pas ouverts.

Ainsi le programme peut utiliser une structure standard d'instructions PLAY, MUSIC et WAIT, et lire tous les paramètres des instructions à partir d'instructions DATA. Ainsi ce programme fait la même chose que le précédent :

```
10 TIME=17
20 RESTORE
30 READ A'NOMBRE TOTAL DE NOTES
40 FOR K=1 TO A
50 READ C1,S1,E1,P1
60 READ L1,T1,N1,V1,L2,T2,N2,V2,L3,T3,N3,V3
70 READ PERIODES
80 READ C2,S2,N2,P2
90 WAIT TIME*R-2
100 PLAY S,T,U,V:WAIT 2
110 IF KEY$<>""THEN PLAY 0,0,0,0:STOP
120 NEXT
130 DATA 4,3,0,0,0,1,3,8,5,2,3,8,5,0,1,0,2,3,0,0,0
140 DATA 7,0,0,0,1,3,12,5,2,3,3,5,3,2,8,5,2,4,0,0,0
150 DATA 7,0,0,0,1,3,12,5,2,3,3,5,3,2,8,5,3,4,0,0,0
160 DATA 7,0,0,0,1,3,12,5,2,3,3,5,4,2,8,5,2,0,0,0,0
```

Ce programme ne paraît pas plus court que le programme précédent, pourtant il est probablement un peu plus rapide à frapper. Si toutefois nous jouions 400 notes au lieu de 4 les avantages de la seconde méthode seraient plutôt évidents. Chaque note ajoutée requiert une ligne supplémentaire de DATA plus une modification du premier élément de la première instruction DATA qui indique le nombre total de notes.

Le nombre d'éléments dans les DATA peut être réduit en mettant des quantités qui restent constantes (mode d'enveloppe, période de l'enveloppe, volume, canal sélectionné) dans les lignes 50 à 100. Tel qu'il est toutefois le programme est plus versatile. Il nous permet de varier le volume de chaque partie de l'air pour tous les canaux ou pour chaque canal. Il nous permet de changer le mode de l'enveloppe, d'introduire éventuellement un trémolo ; il permet d'ajouter du bruit sur n'importe quel canal pour des effets spéciaux.

La ligne 110 permet d'arrêter le programme en appuyant sur n'importe quelle touche.

Adaptez ce programme pour écrire n'importe quel air de votre choix. Expérimentez avec différents effets.

Chapitre 12
ORIC L'ARTISTE

GRAPHIQUES EN BASSE RESOLUTION

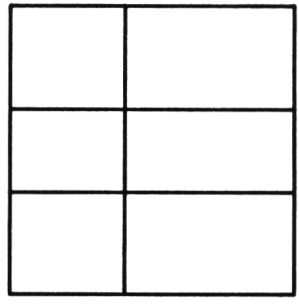
Nous avons déjà vu que la touche ESC suivi par I,K,M ou O, ou CHR (27) suivi par "I","K","M" ou "O", crée des ensembles de caractères alternatifs. Ces caractères alternatifs peuvent être utilisés pour dessiner sur l'écran des caractères à basse résolution dits aussi "en morceaux". Nous avons déjà vu les caractères alternatifs utilisés pour imprimer des formes sur l'écran aux chapitres 1, 7 et 9.

SELECTION DE CARACTERES ALTERNATIFS

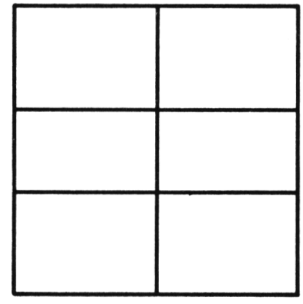
Si vous construisez une image en utilisant des graphiques grossiers vous pouvez sélectionner les caractères dont vous avez besoin pour cette partie de l'image en vous référant à l'annexe A.

Toutefois, il doit y avoir des moyens plus faciles pour construire un caractère que de l'essayer et de le trouver sur une liste. Heureusement la procédure pour faire cela est relativement facile.

Pour générer des caractères alternatifs, ORIC divise la position représentée par le caractère en 6. Ces divisions ne sont pas tout à fait égales.

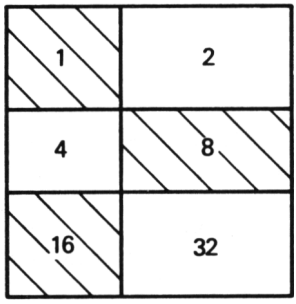


ORIC (V1.0)

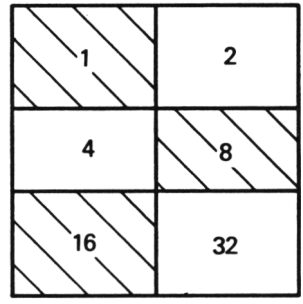


ORIC ATMOS (V1.1)

Les caractères alternatifs sont créés en encrant les zones sélectionnées et en laissant les autres à la couleur du papier. Prenons un exemple :



ORIC (V1.0)



ORIC ATMOS (V1.1)

J'ai numéroté les zones 1,2,4,8,16,32. Pouvez-vous comprendre la signification de ces nombres? Regardez la notation binaire expliquée au chapitre 10 et calculez les équivalents binaires de ces nombres si vous avez besoin d'un indice.

Suivez cette procédure simple pour générer un caractère quelconque que vous désirez :

1. Dessinez le caractère, partagez-le en zones et numérotez les zones comme je l'ai fait.

2. Sélectionnez les zones que vous voulez voir encrées.

3. Additionnez les nombres des zones sélectionnées. Dans l'exemple que j'ai donné, 1+8+16 = 25.

4. Ajoutez 32 à ce nombre (les codes ASCII des caractères normaux commencent à 32). Dans l'exemple donné le résultat serait 25+32 = 57.

Le nombre auquel vous avez abouti sera égal à la valeur du code ASCII pour le caractère alternatif que vous désirez.

Pour imprimer la forme choisie sur l'écran nous pouvons utiliser le code ASCII directement :

```
PRINT CHR$(27)"I"CHR$(57)
```

ou bien nous pouvons regarder dans l'annexe A pour trouver le caractère qui correspond à ce code ASCII. Dans l'exemple choisi le caractère "9" a le code ASCII 57. Ainsi nous pourrions utiliser :

```
PRINT CHR$(27)"I9"
```

Pratiquez en utilisant cette procédure vous verrez que cela devient une routine assez rapidement.

Quand vous imprimerez un certain nombre de caractères fusionnés ensemble pour former une image, vous trouverez probablement qu'il est plus pratique de définir une chaîne de caractères contenant tous ces caractères. Vous pouvez vous souvenir que nous avons fait cela au chapitre 9.

LORES 1

Avec la machine V1.0 il est maladroit de placer un caractère dans une position choisie sur l'écran en choisissant PRINT. Il est plus pratique d'utiliser PLOT.

Il serait aussi pratique de ne pas avoir à utiliser CHR$(27)"I..." pour générer des caractères alternatifs.

ORIC nous permet d'utiliser PLOT et d'oublier CHR$(27)"I" en fournissant un mode d'affichage à l'écran connu sous le nom de LORES 1. En mode LORES 1 tous les caractères sont imprimés normalement comme des caractères alternatifs, et n'ont pas à être spécifiés comme tels.

Nous avons étudié LORES 0 au chapitre 7. LORES 1 est similaire en beaucoup de points mais plutôt plus utile (du moins selon moi).

Pour mettre ORIC en mode LORES 1, utilisez l'instruction LORES 1 aussi bien comme commande immédiate que dans un programme. Cette instruction efface l'écran et met par défaut la couleur du papier à noir et la couleur de l'encre à blanc.

Si vous utilisez CLS ou CTRL L en LORES 1, la machine revient au mode TEXT. Effacez l'écran avec l'instruction LORES 1.

Comme avec LORES 0, vous pouvez utiliser la colonne 0 en LORES 1 à condition que vous ne souhaitiez pas changer la couleur de l'encre.

Le prochain programme génère une forme sur l'écran qui peut (avec un peu d'imagination ou avec la mauvaise paire de lunettes) être prise pour une soucoupe volante. Vous pouvez déplacer la soucoupe volante sur l'écran en utilisant les touches de contrôle du curseur. En appuyant sur la barre d'espace, cela arrête le programme.

Pour déplacer une forme sur un écran nous imprimons des espaces sur la forme supprimée et l'imprimons de nouveau dans une position juste à côté de celle où elle était précédemment.

```
10 LORES 1:PRINT CHR$(17)
20 SOUCOUPE$=CHR$(70)+CHR$(87)+ CHR$(36)
30 X=18:Y=13
40 : REPEAT
50 : PLOT X,Y,SOUCOUPE$
60 : GET A$
70 : IF ASC(A$)=8 AND X 1 THEN DX=-1
80 : IF ASC(A$)=9 AND X 36 THEN DX=1
90 : IF ASC(A$)=11 AND Y 0 THEN DY=-1
100 : IF ASC(A$)=10 AND Y 25 THEN DY=1
110 : PLOT X,Y,"   ":REM TROIS ESPACES
120 : X=X+DX:Y=Y+DY:DX=0:DY=0
130 : UNTIL ASC(A$)=32
140 CLS:PRINT CHR$(17)
```

NOTE - Ce programme fonctionnera avec les machines V1.0 et V1.1 mais comme résultat il n'utilise pas l'écran dans son entier. Si vous avez la machine V1.1 vous pouvez, si vous le souhaitez, utiliser PRINT @ au lieu de PLOT.

Au chapitre 7 nous avons vu que nous pouvions utiliser l'instruction PLOT pour mettre des attributs de couleur sur l'écran. Nous pouvons faire rougir la soucoupe volante quand elle entre dans la partie droite de l'écran, en ajoutant la ligne :

```
125 : IF X>18 THEN PLOT X-1,Y,1
```

Vous pouvez imprimer des caractères normaux en mode LORES 0 si vous souhaitez mélanger du texte et des graphiques. L'attribut 8 vous remettra sur l'ensemble des caractères standards et l'attribut 9 vous remettra sur l'ensemble des caractères alternatifs. Essayez :

```
10 LORES 1
20 PLOT 4,13,8:PLOT 5,13,"STANDARD":PLOT 13,13,9
30 PLOT 14,13,"ALTERNATIF":PLOT 23,13,8
40 PLOT 24,13,"STANDARD".
```

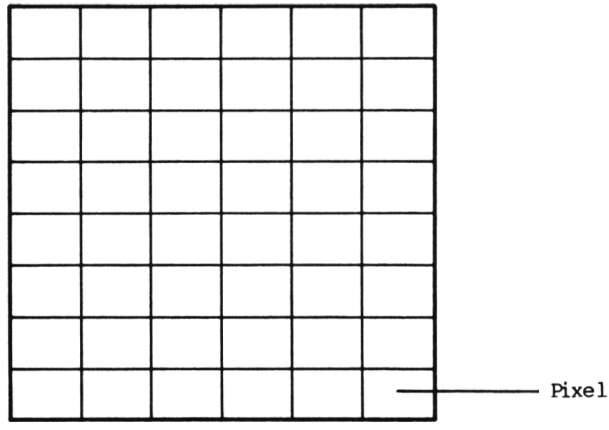
Vous pouvez définir une couleur globale d'encre autre que blanche dans le mode LORES 1 à condition que vous n'utilisiez pas les instructions PRINT ou PLOT dans la colonne 0. Essayer de définir une couleur globale de papier produit un effet bizarre. La couleur de l'arrière-plan peut être changée ligne par ligne en utilisant les attributs d'arrière-plan de l'instruction PLOT (voir chapitre 7).

GRAPHIQUES DEFINIS PAR L'UTILISATEUR

L'ensemble des caractères alternatifs est utile pour des formes grossières mais parfois plus de détails sont nécessaires. Plus de détails, appelés encore haute résolution, peuvent être obtenus en redéfinissant les caractères - c'est-à-dire en changeant la forme des caractères selon la forme que vous voulez.

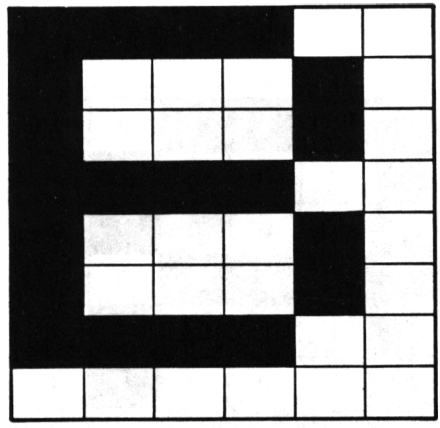
L'image faite sur l'écran de télévision contrôlée par l'ordinateur est faite d'un grand nombre de points tout comme les images dans un journal. Ces points sont appelés les éléments de l'image ou pixels.

Il y a 48 pixels dans chaque position de caractère. Si vous imaginez que chaque position de caractère sur l'écran est divisée en une grille de 8 lignes et de 6 colonnes, alors chaque espace dans la grille est un pixel.



1 Emplacement d'un Caractère

Pour générer un caractère, les pixels sélectionnés sont encrés.



Le schéma de chaque caractère est contenu dans la mémoire de l'ORIC. Chaque caractère prend 8 positions mémoire, une pour chaque rangée de pixels.

Les ordinateurs stockent l'information sous forme de nombres binaires. ORIC se souvient de la forme du caractère en donnant à chaque pixel encré une valeur de bit de 1 et à chaque pixel non encré une valeur de bit de 0.

Ainsi la lettre B est représentée par :

1	1	1	1	0	0
1	0	0	0	1	0
1	0	0	0	1	0
1	1	1	1	0	0
1	0	0	0	1	0
1	0	0	0	1	0
1	1	1	1	0	0
0	0	0	0	0	0

En ajoutant deux zéros devant, cela permet d'obtenir un octet dont le résultat en binaire suit le schéma ci-dessous :

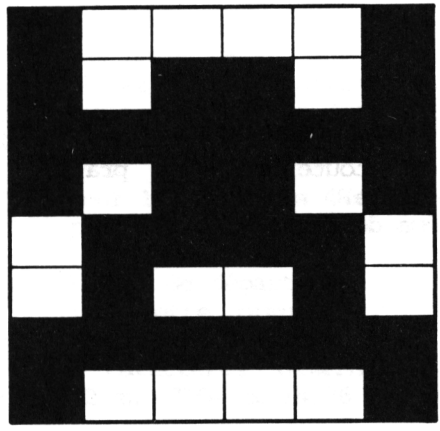
0011 1100
0010 0010
0010 0010
0011 1100
0010 0010
0010 0010
0011 1100
0000 0000

Il est pratique de l'écrire en hexadécimal. Ceci donne :

#3C
#22
#22
#3C
#22
#22
#3C
#00

Pour définir nos propres formes graphiques, nous changeons les nombres contenus dans les positions mémoires où ORIC stocke les formes graphiques.

D'abord choisissez une forme, par exemple :



Ecrivez 1 quand un pixel est encré, 0 quand il ne l'est pas.

1	0	0	0	0	1
	0	1	1	0	1
1	1	1	1	1	1
1	0	1	1	0	1
0	1	1	1	1	0
0	1	0	0	1	0
1		1	1	1	1
1	0	0	0	0	1

Cela vous donne le schéma binaire suivant :

10 0001
10 1101
11 1111
10 1101
01 1110
01 0010
11 1111
10 0001

Vous pouvez convertir ces nombres binaires en valeur décimale. Il est toutefois plus pratique d'utiliser la notation hexadécimale.

Aussi le schéma devient :

#21
#2D
#3F
#2D
#1E
#12
#3F
#21

La prochaine étape est de choisir un caractère qui sera remplacé par la forme que vous venez de concevoir. Faites votre choix prudemment, n'oubliez pas que le caractère gardera sa nouvelle forme jusqu'à ce que vous le changiez de nouveau, ou appuyez sur l'interrupteur RESET pour recharger l'ordinateur.

Par exemple, si vous choisissez de changer R, les instructions telles que PRINT et READ auront une drôle d'allure sur vos listings !

Une possibilité est de changer les caractères tels @ et } , que vous n'utilisez pas souvent. Toutefois si vous avez à changer un nombre important de caractères, il est probablement plus facile de changer les lettres minuscules telles p, q etc... Toutes vos instructions et variables sont en lettres majuscules; aussi cela est suffisamment prudent à condition que vous ne vouliez pas de message en lettres minuscules sur l'écran, et aussi que vous n'utilisiez pas de lettres contenues dans "Ready", "Searching", "Loading" et "Saving". Une troisième pos-sibilité est de changer l'ensemble de caractères alternatifs et d'utiliser le mode LORES 1.

Dans cet exemple nous allons remplacer la lettre "b" par la forme désirée. Pour ce faire nous allons changer l'emplacement mémoire qui contient en ce moment la forme du caractère "b" de telle manière qu'à cette place il y ait les valeurs qui représentent la forme que nous avons choisie. Cela soulève deux problèmes :

1. Comment trouvons-nous l'emplacement mémoire où la forme du caractère "b" est contenue ?

2. Comment changeons-nous le contenu de ces emplacements mémoire ?

Pour trouver l'emplacement où se trouve la forme du caractère, nous utilisons l'une des deux formules (selon que l'ordinateur est de 16k ou de 48k).

Dans les modes TEXT et LORES, l'adresse du premier des 8 emplacements mémoire qui contient la forme du caractère "b" est donnée par :

46080+8*ASC("b") pour les machines 48k.
ou 13312+8*ASC("b") pour les machines 16k.

Pour obtenir l'adresse de départ de toute autre forme de caractère, vous utilisez cette formule, mais n'oubliez pas de mettre le caractère recherché à l'intérieur des guillemets. Pour redéfinir des caractères alternatifs, changez les nombres 46080 et 13312 en 47104 et 14336 respectivement.

POKE et PEEK

Pour changer le contenu d'un emplacement mémoire utilisez l'instruction POKE. Le format est :

POKE ADRESSE, NOMBRE

où ADRESSE est égale à l'adresse de l'emplacement mémoire que vous souhaitez changer, et NOMBRE est la valeur que vous souhaitez mettre dans cette adresse.

PEEK est l'opposé de POKE. Il nous permet de trouver quel est le nombre qui est stocké dans un emplacement mémoire.

PRINT PEEK (14000)

imprimera sur l'écran le nombre qui est stocké dans l'adresse mémoire 14000. La lecture d'une adresse par l'instruction PEEK ne change pas le contenu de cet emplacement mémoire.

Dans ce cas, nous voulons changer le contenu de l'emplacement mémoire, de manière que POKE soit l'instruction à utiliser. On va introduire par l'instruction POKE des nombres dans les 8 emplacements mémoire qui vont représenter la forme choisie qui contient à présent la forme du caractère b.

Si vous avez une machine 16k tapez :

```
10 A=13312+8*ASC("b")
```

Si vous avez une machine 48k tapez :

```
10 A=46080+8*ASC("b")
```

Le reste du programme est le même pour les deux machines :

```
20 FOR N=0 TO 7
30 READ B
40 POKE A+N,B
50 NEXT
60 DATA #21,#2D,#3F,#2D,#1E,#12,#3F,#21
```

Exécutez ce programme. Enlevez capitals lock et appuyez sur la touche B. Le caractère "b" a été remplacé par la forme que vous avez définie.

Listez votre programme. Même sur le listing du programme, "b" a été altéré. En appuyant sur l'interrupteur RESET vous restaurerez "b" à sa forme normale.

NOTE - Dans la machine V1.0 l'instruction POKE n'acceptera pas un nombre hexadécimal comme second paramètre - c'est-à-dire vous ne pouvez écrire POKE 16000,#21. Par contre vous pouvez faire A=#21:POKE 16000,A. Les deux paramètres peuvent être en hexadécimal pour la machine V1.1.

ANIMATION

Vous pouvez déplacer un caractère sur l'écran en imprimant un espace sur le caractère et en imprimant le caractère dans une position adjacente. Une autre méthode pour simuler le mouvement est d'imprimer un caractère sur l'écran et de surimprimer dessus un autre caractère qui est seulement légèrement différent. Le programme suivant illustre cela. Si vous avez une machine 16k changez 46080 en ligne 10 par 13312.

```
10 A=46080+8*ASC("p")
20 : FOR K=0 TO 8 STEP 8
30 :  FOR N=0 TO 7
40 :  READ B
50 :  POKE A+N+K,B
60 : NEXT N,K
70 CLS
80 A$="p q p q":B$="p q p q"
90 : REPEAT
100 : PLOT 13,13,A$:WAIT 50
110 : PLOT 13,13,B$:WAIT 50
120 : UNTIL KEY$<>"":REM N'IMPORTE QUELLE TOUCHE
130 END'ARRETE LE PROGRAMME
140 DATA #21,#2D,#3F,#2D,#1E,#12,#3F,#21
150 DATA #00,#0C,#1E,#2D,#3F,#21
```

SCRN

Les graphiques définis par l'utilisateur sont très utilisés dans les jeux sur ordinateur. Une autre fonction utile pour une telle application est SCRN. SCRN est de la forme :

```
SCRN(X,Y)
```

Cette instruction donne la valeur ASCII du caractère à la position X,Y sur l'écran. C'est particulièrement utile si vous voulez prendre une action spéciale quand un caractère qui se déplace sur l'écran atteint une position particulière.

Dans le programme suivant des particules de matière (représentées par M) et d'anti-matière (représentées par A) apparaissent dans des positions aléatoires sur l'écran. Si la matière et l'anti-matière apparaissent à la même place, il y a explosion.

```
10 REPEAT
20 CLS:PAPER 7:INK 4
30 : FOR N=1 TO 20
40 : X%=2+37*RND(1)
50 : Y%=26*RND(1)
60 : PLOT X%,Y%,"M"
70 : NEXT
80 WAIT 50
90 : FOR N=1 TO 20
100 : X%=2+37*RND(1)
110 : Y%=26*RND(1)
120 : IF CRN(X%,Y%)=ASC("M") THEN 150 ELSE PLOT X%,Y%,"A"
130 : NEXT
140 UNTIL 0
150 EXPLODE
160 : FOR N=0 TO 10
170 : PAPER 1:WAIT 5:PAPER 7:WAIT 5
180 : NEXT
190 INK 0:CLS:END
```

Chapitre 13

CARACTERISTIQUES ARTISTIQUES SUPPLEMENTAIRES D'ORIC

HIRES

Quand nous avons étudié les graphiques définis par l'utilisateur nous avons vu que chaque position de caractère peut être divisée en 48 pixels. ORIC possède un mode graphique haute résolution qui vous permet de travailler avec les pixels à l'écran au lieu de travailler avec les caractères, de telle manière que vous puissiez dessiner des lignes plus fines et avoir des dessins plus détaillés.

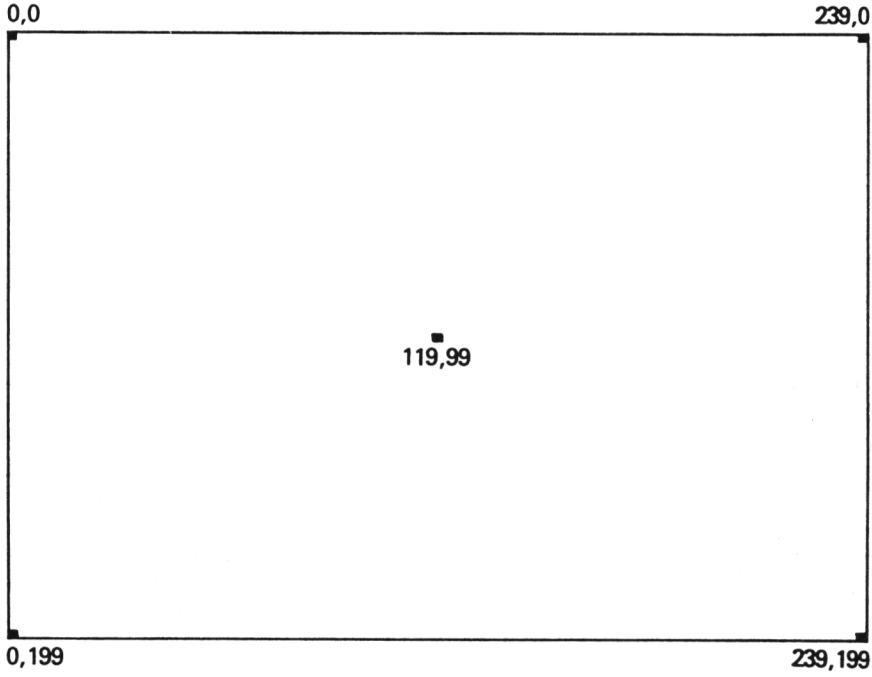
L'instruction HIRES (abrégé de High Resolution) met ORIC dans le mode haute résolution. HIRES peut être utilisé comme commande immédiate ou comme une instruction de programme.

Quand ORIC passe en mode haute résolution l'écran devient noir à part les trois lignes du bas qui restent en mode basse résolution et peuvent être utilisées pour entrer du texte.

ECRAN HAUTE RESOLUTION

L'écran haute résolution est composé d'un ensemble de 48000 pixels - l'écran comporte donc 240 pixels en largeur et 200 pixels en hauteur. La distance sur la largeur de l'écran est appelée la coordonnée X. La distance sur la hauteur de l'écran est appelée la coordonnée Y.

Le diagramme suivant montre la position de plusieurs points importants sur l'écran.



Le curseur graphique est constitué d'un seul pixel. Contrairement au curseur de texte, il ne clignote pas car cela pourrait gâcher votre affichage. Le curseur graphique est lui-même invisible. Toutefois il est possible d'encrer le pixel sur lequel le curseur graphique est positionné actuellement.

Quand la machine passe en mode haute résolution ou à chaque fois que l'instruction HIRES est utilisée, le curseur graphique est positionné à l'emplacement 00.

L'instruction HIRES efface la partie haute résolution de l'écran. CLS fonctionne seulement sur la partie basse résolution au bas de l'écran.

CURSET

L'instruction CURSET déplace le curseur graphique vers un pixel spécifique. Le format est :

```
CURSET X,Y,FB
```

où X et Y sont les coordonnées du pixel, et FB est un nombre qui détermine ce que l'instruction CURSET fait au pixel, une fois que le curseur y est.

Si FB vaut 0, le pixel prend la couleur de l'arrière-plan (papier).

Si FB vaut 1, le pixel prend la couleur de l'avant-plan (encre).

Si FB vaut 2, la couleur du pixel est inversée. Si il est originellement de la couleur de l'avant-plan, il prend la couleur de l'arrière-plan et vice-versa.

Si FB vaut 3, la couleur du pixel est inchangée.

Le programme suivant imprime des points à des positions aléatoires sur l'écran. Si la même position est sélectionnée deux fois, le point disparaît.

```
10 REM MALADIE BIZARRE
20 HIRES
30 REPEAT
40 X%=240*RND(1):Y%=200*RND(1)
50 CURSET X%,Y%,2
60 UNTIL KEY$<>"":REM
70 TEXT:END'N'IMPORTE QUELLE TOUCHE ARRETE LE PROGRAMME
   ET REMET EN MODE TEXTE.
```

CURMOV

Quelquefois vous pouvez souhaiter déplacer le curseur d'un certain nombre de pixels dans les directions X et/ou Y. Vous pourriez calculer la position courante du curseur (mais ce n'est pas toujours facile), puis calculer à partir de cela l'adresse actuelle requise par la nouvelle position du curseur. Toutefois ceci est plutôt laborieux.

CURMOV fait tout cela pour vous. Vous spécifiez le nombre de pixels dans chaque direction du déplacement du curseur que vous souhaitez et ORIC fait le calcul. Le format est :

CURMOV X,Y,FB

où X et Y spécifient le mouvement relatif de la position courante du curseur et où FB joue le même rôle que précédemment. Dans cette instruction X et Y peuvent être positifs et négatifs. Le programme suivant le montre :

```
10 HIRES
20 CURSET 69,49,3
30 FOR COTE=100 TO 20 STEP -10
40 CURMOV COTE,0,1
50 CURMOV 0, COTE,1
60 CURMOV -COTE, 0,1
70 CURMOV 0,-COTE,1
80 CURMOV 5,5,3
90 NEXT
100 GET A$
110 TEXT:END' N'IMPORTE QUELLE TOUCHE ARRETE LE PROGRAMME
    ET REPASSE EN MODE TEXTE.
```

DRAW

L'instruction DRAW trace une ligne droite de la position courante du curseur à la nouvelle position du curseur. La nouvelle position du curseur est spécifiée de la même manière que pour l'instruction CURMOV, c'est-à-dire par un déplacement relatif à partir de la position courante du curseur. Le format est :

DRAW X,Y,FB

où X et Y spécifient la différence des coordonnées X et Y entre la position initiale du curseur et la position finale du curseur. X et Y peuvent être positifs ou négatifs.

Si FB=0 la ligne est tirée dans la couleur de l'arrière-plan. Ceci est principalement utilisé pour supprimer les lignes. Pour s'assurer de l'effacement total, les lignes doivent être supprimées dans la même direction que celle où elles ont été tracées.

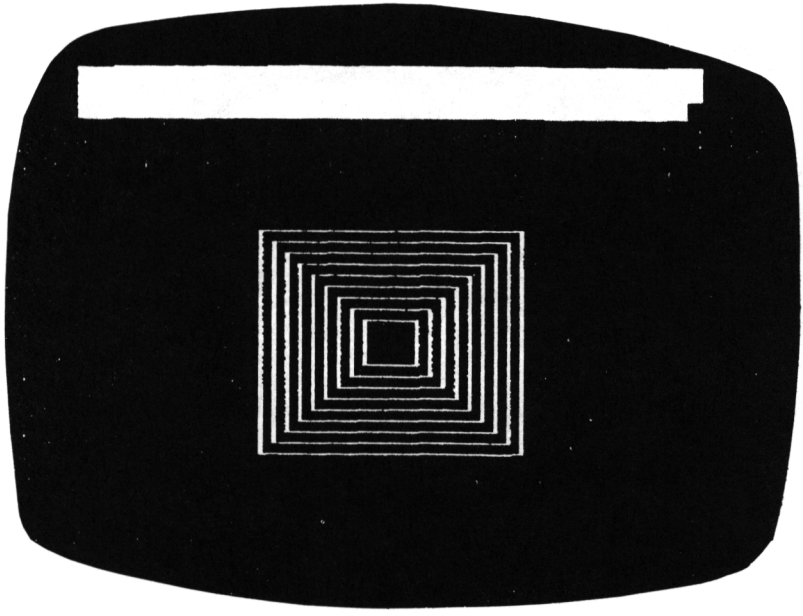
Si FB=1 la ligne est tirée dans la couleur de l'avant-plan.

Si FB=2 alors tous les pixels dans la ligne ayant présentement la couleur de l'avant-plan prennent la couleur de l'arrière-plan. Tous les pixels ayant la couleur de l'arrière-plan prennent la couleur de l'avant-plan. Des effets très intéressants peuvent être obtenus par ce moyen.

Si FB=3 le curseur se déplace sans affecter l'affichage à l'écran. Ceci n'est pas utilisé très souvent car CURMOV fait la même chose.

Nous pouvons adapter le programme précédent de telle manière qu'il tire des lignes entre les points :

```
10 HIRES
20 CURSET 69,49,3
30 FOR COTE:100 TO 20 STEP -1
40 DRAW COTE 0,1
50 DRAW 0,COTE,1
60 DRAW -COTE,0,1
70 DRAW 0,-COTE,1
80 CURMOV 5,5,3
90 NEXT
100 GET A$
110 TEXT:END"N'IMPORTE QUELLE TOUCHE ARRETE LE PROGRAMME
    ET REPASSE EN MODE TEXTE.
```

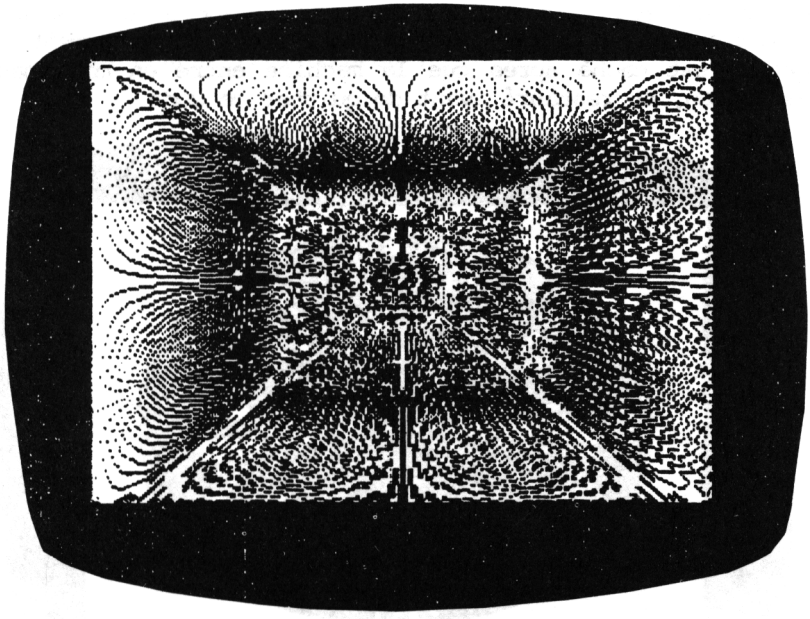


Ce programme peut être encore modifié de telle manière qu'il efface chaque carré avant de dessiner le suivant :

```
10 HIRES
20 REPEAT
30 CURSET 69,49,3
40 : FOR COTE=100 TO 20 STEP -10
50 : FOR N=1 TO 0 STEP -1
60 : DRAW COTE,0,N
70 : DRAW 0,COTE,N
80 : DRAW 0,-COTE,0,N
90 : DRAW 0,-COTE,N
100 : WAIT 50 N
110 : NEXT N
120 : CURMOV 5,5,3
130 : NEXT COTE
140 : UNTIL KEY$<>"":REM PAS D'ESPACE
150 : TEXT:END
```

En utilisant l'instruction DRAW avec FB=2, on obtient ce qui est appelé des "franges d'interférence" entre lignes adjacentes. Essayez :

```
10 PAPER 0:HIRES:PRINT CHR$(17)
20 FOR X:0 TO 239
30 CURSET X,0,3
40 DRAW 239-2*X,199,2
50 NEXT X
60 FOR Y=0 TO 199
70 CURSET 0,Y,3
80 DRAW 239,199-2*Y,2
90 NEXT Y
100 GET A$
110 PRINT CHR$(17):TEXT:PAPER 7
```



COULEUR GLOBALE DANS LE MODE HAUTE RESOLUTION

Vous pouvez utiliser la totalité de la surface de la portion de l'écran consacrée à la haute résolution si vous souhaitez dessiner en blanc sur un fond noir. Vous pouvez changer les couleurs globales du papier et de l'encre, mais si vous le faites vous perdrez tout ce qui est dessiné sur les deux premières colonnes de l'écran. Cela vous empêchera d'encrer tout pixel dont la coordonnée X est inférieure à 12.

Pour illustrer ceci, ajoutez au programme précédent :

```
95 PAPER 4:INK 3
```

Il y a encore deux points intéressants dans ce programme. D'abord la partie basse résolution de l'écran est positionnée à une couleur de papier noire avant de rentrer dans le mode haute résolution, ceci pour donner un arrière-plan entièrement noir. La couleur de papier blanche est restaurée en entrant de nouveau dans le mode TEXT. Deuxièmement, le curseur est caché après que la machine entre en mode haute résolution, et restauré avant de quitter ce mode.

CIRCLE

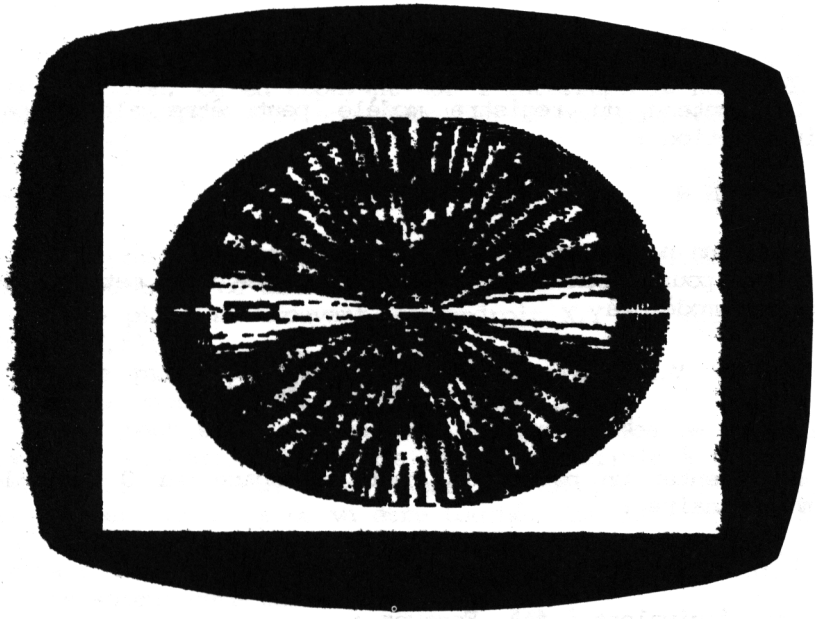
L'instruction CIRCLE trace un cercle dont le centre est la position courante du curseur. Le format est :

CIRCLE R,FB

Où R est le rayon et FB est identique à celui de l'instruction DRAW.

Vous pouvez aussi tracer des schémas avec interférences à l'aide de l'instruction CIRCLE :

```
10 PAPER 0:HIRES:PRINT CHR$(17)
20 PAPER 6:INK 1
30 FOR R=2 TO 89 STEP 2
40 CURSET 110,100,3:CIRCLE R,2
50 CURSET 130,100,3:CIRCLE R,2
60 NEXT
70 GET A$
80 PRINT CHR$(17):TEXT:PAPER 7
```



Note - Quand vous utilisez les instructions CIRCLE, DRAW et CURMOV faites attention à ne pas spécifier de nombre qui fassent sortir des limites de l'écran. Si vous essayez de dessiner en dehors de l'écran, vous obtiendrez des erreurs.

PATTERN

ORIC possède un dispositif spécial qui vous permet de tirer des lignes pointillées. Ceci est contrôlé par un emplacement mémoire de 8 bits appelé le **registre modèle.**

Dans le registre modèle, un 1 vous donne 1 point et un 0 un espace. Quand la machine est mise sous tension tous les bits dans le registre sont positionnés à 1. Ce qui donne des lignes continues :

Le nombre binaire stocké dans le registre modèle, lors de la mise sous tension est donc :

1111 1111

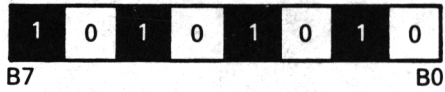
ce qui est égal à #FF ou 255 (décimal).

Le contenu du registre modèle peut être altéré par l'instruction :

PATTERN N

où N est un nombre entre 0 et 255 (#00 et #FF).

Ainsi pour obtenir une ligne pointillée nous mettrons le registre modèle à



Représenter un point par 1 et un espace par 0 donne le nombre binaire :

1010 1010

qui est équivalent à #AA. Essayez :

```
10 HIRES
20 PATTERN #AA
30 DRAW 239,199,1
40 GET A$:TEXT
```

Des pointillés peuvent être obtenus en mettant le registre modèle à

1	1	0	0	1	1	0	0

Ce qui donne 1100 1100 (en binaire) ou #CC.
Essayez :

```
10 HIRES
20 PATTERN #CC
30 CURSET 120,100,3
40 CIRCLE 90,1
50 NEXT A$:TEXT
```

Les autres nombres que vous pouvez essayer avec l'instruction PATTERN sont :

1111 0000 (en binaire) ou #F0 (pointillés plus longs)
1111 1000 (en binaire) ou #F8
1111 1010 (en binaire) ou #FA

En coupant l'alimentation de l'ordinateur et en remettant sous tension, ou en appuyant sur l'interrupteur RESET ou en utilisant PATTERN #FF, vous remettez le registre modèle à son état initial.

POINT

L'instruction POINT est utilisée pour déterminer si un pixel est encré ou non.

Si le pixel se trouvant à l'endroit X,Y est encré, alors POINT (X,Y) = -1.

Si le pixel se trouvant à l'emplacement X,Y de l'écran n'est pas encré, alors POINT (X,Y) = 0.

POINT peut être utilisé dans les programmes de jeux. Par exemple, si vous déplacez une forme pré-définie à travers un labyrinthe de points et de lignes, POINT pourrait vous dire si un obstacle a été ou va être touché. Une autre utilisation de l'instruction POINT consiste à remplir des zones en encrant les pixels jusqu'à ce que des pixels préalablement encrés soient détectés.

Le programme suivant montre cela :

```
10 HIRES
20 CURSET 20,100,1
30 DRAW 70,-50,1
40 DRAW 0,70,1
50 FOR X=20 TO 89
60 Y=120
70 REPEAT
80 CURSET X,Y,1
90 Y=Y-1
100 UNTIL POINT(X,Y)=-1
110 NEXT
120 GET A$:TEXT
```

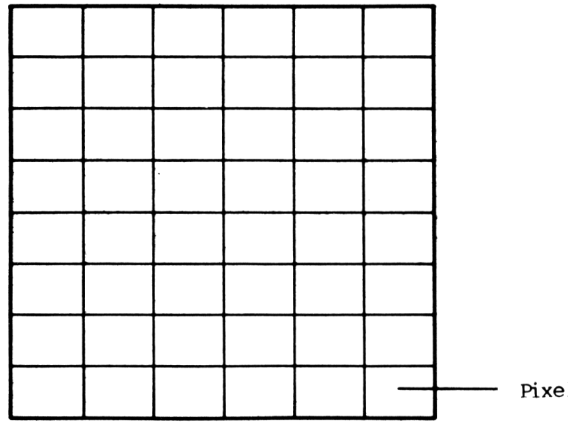
FILL

La méthode décrite pour remplir une zone fonctionne bien avec des formes petites ou irrégulières. Toutefois elle est lente. L'instruction FILL permet de garnir de larges zones rectangulaires de l'écran très rapidement. Le format est :

```
FILL X,Y,N
```

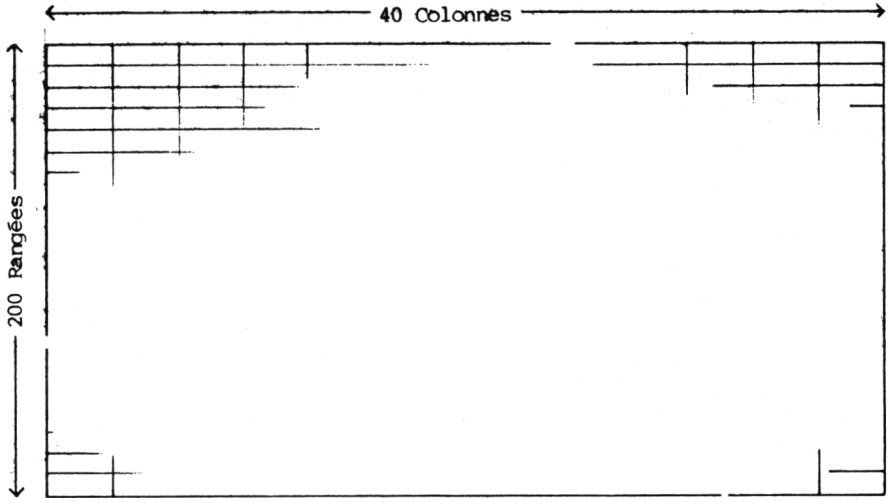
où Y,X et N sont tous des nombres. Y et X définissent les dimensions de la zone à remplir et N définit le modèle binaire qui remplira cette zone. Regardons la manière dont l'écran est divisé.

Vous vous rappelez que chaque emplacement contenant un caractère est divisé en un tableau de 6x8 pixels :



1 Emplacement d'un Caractère

Pour les besoins de l'instruction FILL, 6 pixels horizontaux consécutifs peuvent être groupés en un bloc. Ainsi, la division de l'écran devient :



L'instruction FILL remplit Y rangées ayant chacune X blocs de large avec un modèle défini par N. N définit le modèle binaire de chaque bloc, si bien que si les 6 pixels d'un bloc doivent être encrés, N serait égal à 111111 (binaire) ou #3F ou 63 (décimal).

Essayez :

```
10 HIRES
20 FILL 16,4,#3F
```

Ceci remplit 16 rangées (chacune ayant 1 pixel d'épaisseur) sur 4 colonnes (chacune ayant 6 pixels de large) dans le coin supérieur gauche de l'écran.

Essayez :

```
10 HIRES
20 FILL 16,4,#2A
```

Celui-ci remplit 4 blocs espacés également, remplissant ainsi la même zone que précédemment avec des lignes verticales.

Y est un nombre entier compris entre 1 et 199. X est un nombre entier compris entre 1 et 40. Afin de faciliter les opérations de remplissage, ORIC fait l'hypothèse qu'au moins un pixel (le premier) dans le bloc est encré. Aussi pour le remplissage normal N est un nombre entier compris entre 100000 (binaire) et 111111 (binaire) ou #20 et #3F, ou 32 et 63.

La position de départ du bloc à remplir est définie par la position du curseur graphique lorsque l'instruction FILL est exécutée. La position du curseur graphique au début du remplissage définit la position du coin supérieur gauche du bloc. Dans la machine V1.1 lorsque le remplissage est fini le curseur graphique se trouve dans le coin inférieur gauche du bloc prêt à commencer un nouveau bloc directement en dessous si nécessaire. Dans la machine V1.0 la position du curseur après l'instruction FILL n'est pas définie. Utilisez CURSET pour redéfinir la position du curseur dans ce cas.

Essayez :

```
10 HIRES
20 CURSET 108,92,3
30 FILL 16,4,#3F
```

A moins que nous déplacions de nouveau le curseur la prochaine opération de remplissage commencera où la dernière l'a laissé (dans la machine V1.1).

Essayez :

```
10 HIRES
20 CURSET 84,52,3
30 FILL 32,4,#2A
40 FILL 32,4,#3F
50 FILL 32,4,#38
```

Ajoutez les lignes :

```
35 CURSET 116,52,3
45 CURSET 148,52,3
```

dans la machine V1.0.

ATTRIBUTS EN MODE HAUTE RESOLUTION

Si le troisième nombre de l'instruction FILL est inférieur à 32 (#20) ORIC traite le nombre comme un attribut qui affectera tous les blocs qui le suivent sur la même rangée sur laquelle il est placé.

Les attributs associés à chaque nombre sont similaires à ceux du mode basse résolution, c'est-à-dire :

0 - encre noire	16 - papier noir
1 - encre rouge	17 - papier rouge
2 - encre verte	18 - papier vert
3 - encre jaune	19 - papier jaune
4 - encre bleue	20 - papier bleu
5 - encre magenta	21 - papier magenta
6 - encre cyan	22 - papier cyan
7 - encre blanche	23 - papier blanc

Les attributs des numéros 12 à 15 font que les couleurs d'avant-plan clignotent.

Vous pouvez avoir toutes les couleurs d'arrière-plan et d'avant-plan à l'intérieur d'un seul emplacement de caractère.

Essayez :

```
10 HIRES
20 CURSET 117,96,3
30 FILL 8,1,33
40 CURSET 105,96,3
50 FOR N=0 TO 7
60 FILL 1,1,N
70 NEXT
80 CURSET 129,88,3
90 FILL 24,1,6
100 REPEAT
110 : FOR N=16 TO 23
120 : CURSET 111,88,3
130 : FILL 24,1,N
140 : WAIT 50
150 : NEXT
160 : UNTIL KEY$<>"":REM PAS D'ESPACE
170 : TEXT:END
```

Qui a dit qu'ORIC ne pouvait utiliser que deux couleurs en mode haute résolution ?

Cet exemple est bien meilleur sur une grande surface. Le but de ce dernier programme est de démontrer la finesse avec laquelle on peut définir les couleurs.

Pour voir les effets des couleurs d'avant-plan clignotantes, ajoutez les lignes :

```
95 CURSET 95,96,3
96 FILL 8,1,15
```

LES ATTRIBUTS DE L'INSTRUCTION POKE

Chaque bloc de 6 pixels sur l'écran en mode haute résolution et chaque emplacement de caractère en mode basse résolution, est contrôlé par un emplacement en mémoire. En mettant dans cet emplacement mémoire un nombre compris entre 0 et 31 à l'aide de l'instruction POKE, nous pouvons mettre un attribut directement sur l'écran.

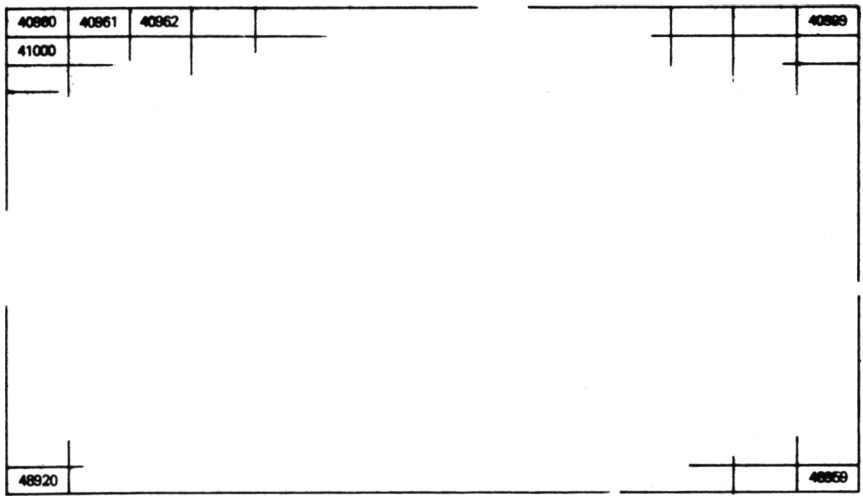
Cela ne présente pas d'avantage supplémentaire dans le mode basse résolution, où l'instruction PLOT est plus facile à utiliser, car vous n'avez pas à calculer l'emplacement mémoire correspondant à une position du caractère sur l'écran.

Toutefois en mode haute résolution l'utilisation de l'instruction POKE évite d'avoir à déplacer le curseur graphique pour sélectionner le bloc approprié. Si vous avez à mettre beaucoup d'attributs sur l'écran, l'utilisation de POKE peut permettre d'économiser de l'espace mémoire.

L'écran haute résolution contient 200 lignes de chacune 40 blocs. 8000 positions mémoire sont donc nécessaires pour contrôler l'écran.

Elles sont numérotées de 40960 à 48959 dans la machine ORIC 48k et de 8192 à 16191 dans la machine 16k. Le bloc dont le numéro est le plus bas correspond au coin supérieur gauche. Le bloc immédiatement adjacent sur la droite, sur la même rangée aura un nombre supérieur de 1 et ainsi de suite tout au long de la rangée. Le bloc au début de la rangée suivante (c'est-à-dire inférieure) a un nombre qui est supérieur de 40 au bloc immédiatement au-dessus, et ainsi de suite.

Ainsi pour l'ORIC 48k nous avons :



Mémoire de l'Ecran en Mode Haute Résolution pour l'ORIC 48K

SUGGESTION -Vous pouvez trouver qu'il est plus facile de se souvenir de l'adresse de début de la mémoire haute résolution en hexadécimal. C'est #A000 pour la machine 48k et #2000 pour la machine 16k.

CHAR

En mode basse résolution vous avez affiché les caractères à l'écran en utilisant les instructions PRINT, PLOT et INPUT. Toutefois en mode haute résolution, INPUT et PRINT ne fonctionnent que sur la partie basse résolution de l'écran en bas de celui-ci (et PRINT entraîne en général un défilement de cette partie de l'écran). PLOT entraîne un message d'erreur dans ce mode : "DISP TYPE MISMATCH ERROR".

Pour afficher les caractères sur l'écran en mode haute résolution nous utilisons l'instruction CHAR. Ceci a le format :

```
CHAR,A,M,FB
```

où A,M et FB sont des nombres.

A est un nombre entier entre 32 et 128 et est égal au code ASCII du caractère à imprimer.

M est égal à 0 pour un caractère standard et à 1 pour un caractère alternatif.

Si FB=0 le caractère est de la couleur de l'avant-plan, et si FB=1 le caractère est de la couleur de l'arrière-plan. Si FB=2 les pixels actuellement de la couleur de l'avant-plan sont mis à la couleur de l'arrière-plan et vice-versa. Cela peut être utiisé pour construire des caractères composés. Si FB est égal à 3 l'instruction n'a pas d'effet.

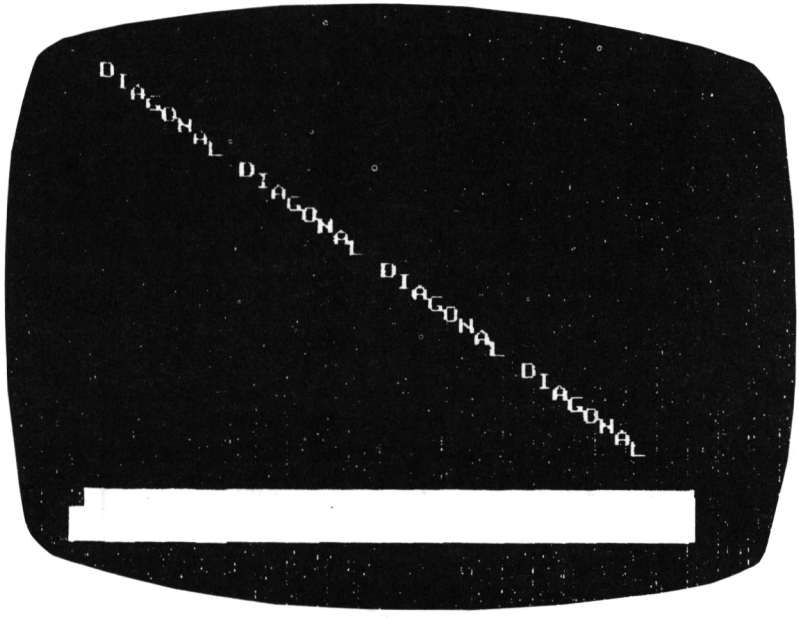
Essayez :

```
HIRES:CHAR 65,0,1
HIRES:CHAR 65,1,1
HIRES:CHAR 127,0,1:CHAR 65,0,0
```

L'instruction CHAR n'affiche qu'un caractère à la fois sur l'écran. La position du caractère sur l'écran est déterminée par la position du curseur graphique. L'instruction CHAR ne déplace pas le curseur graphique, aussi vous devez le déplacer avec CURMOV ou CURSET pour créer une chaîne de caractères.

Essayez :

```
10 HIRES
20 CURSET 12,8,3
30 FOR N=1 TO 4
40 : FOR K=1 TO 9
50 : READ A
60 : CHAR A,0,1
70 : CURMOV 6,5,3
80 : NEXT
90 : RESTORE
100 NEXT
110 REPEAT
120 : FOR N=40960 TO 48962 STEP 40
130 : REM 8192 A 16152 DANS L'ORIC 16k
140 : POKE N,INT(7*RND(1))+1
150 : NEXT
160 UNTIL KEY$<>"":REM PAS DE CARACTERE
170 TEXT:END
180 DATA 68,73,65,71,79,78,65,76,32
```



CARACTERES DEFINIS PAR L'UTILISATEUR

Si vous comparez les emplacements mémoire utilisés pour définir l'écran haute résolution avec ceux utilisés par l'ensemble des caractères définis en mode basse résolution, vous verrez que les deux se superposent. Il s'ensuit que ORIC stocke les caractères autre part dans le mode haute résolution.

En fait, l'espace mémoire mis de côté pour l'ensemble des caractères normaux commence à l'adresse 38192 (adresse 6144 pour la machine 16k) et pour l'ensemble des caractères alternatifs à l'adresse 40836 (7168) ceci dans le mode haute résolution.

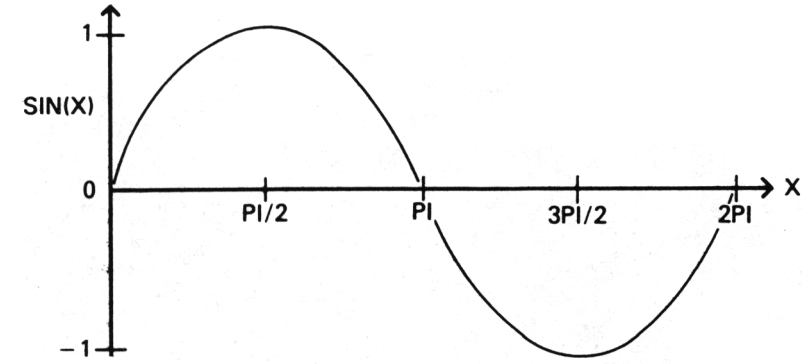
Il est cependant plutôt malcommode d'avoir à se souvenir de deux ensembles d'emplacements. Si vous redéfinissez un caractère en mode basse résolution, il restera redéfini en mode haute résolution. Aussi, quoique vous puissiez redéfinir des caractères en mode haute résolution, il est préférable (du moins je le pense) de faire toutes vos définitions de caractères avant de rentrer dans le mode haute résolution.

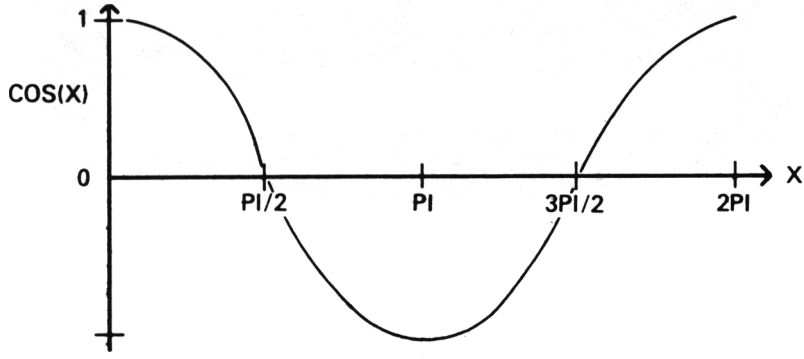
DESSINS DE COURBES

Il est utile de pouvoir dessiner les courbes sur l'écran, à la fois pour construire des images pour des affichages graphiques, et pour fournir des schémas curvilignes pour que des caractères graphiques les suivent lors de programmes de jeux.

Les courbes sont décrites en utilisant des fonctions mathématiques, aussi je dois malheureusement introduire ici un peu de mathématiques. Si cela vous ennuie, ignorez le reste de ce chapitre.

Deux fonctions utiles pour décrire des courbes sont SIN et COS. Elles donnent ceci :

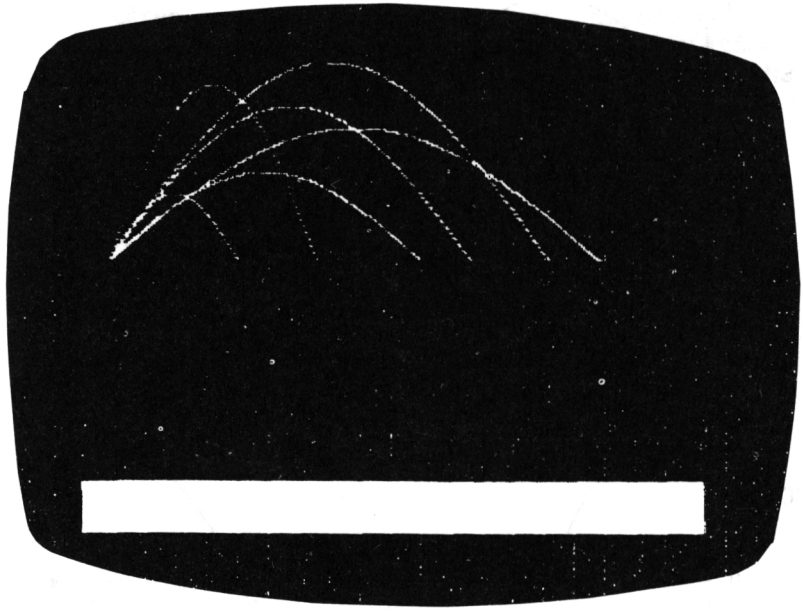




Essayez :

```
10 HIRES
20 DEF FNCURVE(X)=B*SIN(X*PI/A)
30 REPEAT
40 INPUT "LONGUEUR";A
50 INPUT "HAUTEUR";B
60 : FOR X = 1 TO A
70 : CURSET 10+x,100-FNCURVE(X),1
80 : NEXT
90 UNTIL KEY$<>"":TEXT:END
```

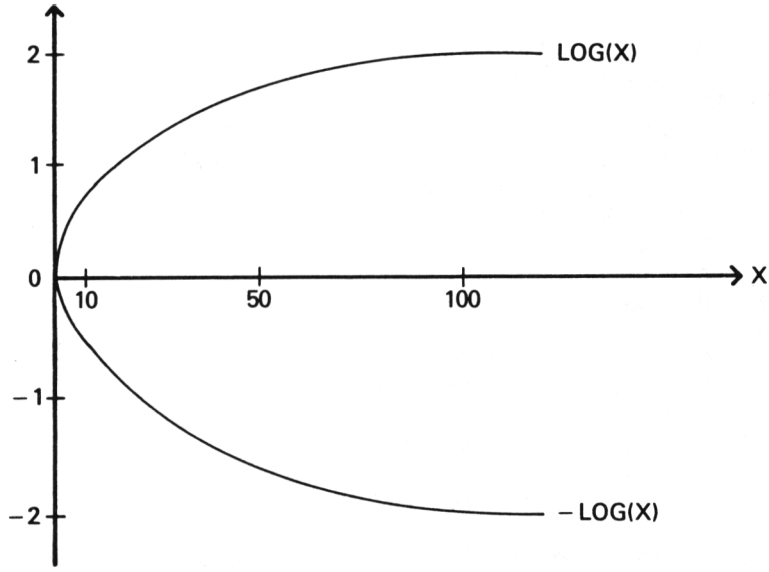
La hauteur entre 20 et 90 (entrée à la ligne 50) et la longueur entre 30 et 220 (entrée à la ligne 40) vous donneront un ensemble de trajectoires.



Essayez l'effet provoqué par le changement de la ligne 20 en :

```
20 DEF FNCURVE(X)=B*X*SIN(X*8*PI/A)/A
```

LOG(X) peut être aussi utilisé.



Ceci peut donner une trajectoire d'un OVNI décollant. -LOG(X) pourrait donner la trajectoire d'un OVNI qui descend rapidement puis plannant au ras du sol.

Par exemple :

```
10 FOR N=1 TO 7
20 READ A
30 POKE 46080+ASC("b")+N,A
40 REM 13312 DANS L'ORIC 16k
50 NEXT
60 HIRES
70 FOR N=0 TO 199
80 FILL 1,1,INT(7*RND(1))+1
90 NEXT
100 REPEAT
110 IF RND(1)>0.5 THEN GOSUB 1000 ELSE GOSUB 2000
120 FOR X=1 TO 211 STEP 6
130 CURSET X+11,FNUFO(X),3
140 CHAR ASC("b"),0,1
150 WAIT 10
160 CHAR ASC("b"),0,0
170 NEXT
180 UNTIL KEYS<>"":TEXT:END
1000 DEF FNUFO(X)=100+X*60*SIN(X*8*PI/210)/210
1010 RETURN
2000 DEF FNUFO(X)=LOG(4*X)*60
2010 RETURN
3000 DATA #38,#1C,#01,#07,#07,#0E,#1C,#38.
```

Assurez-vous que vous tapez ce programme et que vous le lancez initialement en mode TEXT. Cela prendra quelques secondes avant que quelque chose apparaisse. J'espère que cela vaut l'attente.

Chapitre 14

QUELQUES EXPLICATIONS SUPPLEMENTAIRES

MANIPULATION DES DONNEES

Nous pouvons utiliser les instructions DATA pour stocker des données dans un programme. Toutefois si nous souhaitons faire quelque chose avec ceci, nous ferions mieux de les mettre dans un tableau.

Le programme suivant montre comment un tableau de chaîne de caractères à deux dimensions peut être garni de noms et de numéros de téléphone à partir du clavier : et comment les noms commençant par une lettre quelconque pré-selectionnée peuvent être affichés à l'écran accompagnés du numéro de téléphone correspondant.

```
10 PRINT CHR$(12)
20 DIM A$(20,1)
30 FOR N=0 TO 20
40 INPUT"NOM";A$(N,0)
50 IF A$(N,0)="EOF" THEN 90
60 INPUT "NUMERO DE TELEPHONE";A$(N,1)
70 NEXT
80 REM##########################
90 PRINT CHR$(12)
100 INPUT "PREMIERE LETTRE DU NOM"F$:PRINT
110 FOR N=0 TO 20
120 IF A$(N,0)="EOF" THEN 100
130 IF LEFT$(A$(N,0),1)=F$ THEN PRINTA$(N,0)"TELEPHONE"
    A$(N,1):PRINT
140 NEXT:END
```

GESTION DE FICHIERS - STORE ET RECALL

Il est utile de pouvoir stocker sur cassette les données dont nous avons garni un tableau de telle manière que nous puissions les utiliser à nouveau dans le même programme ou dans d'autres. Ceci peut être fait avec la machine V1.1 à condition que le tableau ait été dimensionné préalablement.

NOTE - Si vous allez stocker sur cassette un tableau, il doit être dimensionné, même s'il a moins de 11 éléments. N'oubliez pas que l'instruction DIM efface tous les éléments du tableau - aussi dimensionnez le tableau avant de le garnir.

Les informations contenues dans un tableau sont stockées sur bande dans ce qui est appelé un **fichier**. La première étape pour créer un fichier est de lui choisir un nom. Ici nous allons sauvegarder le tableau A$ dans un fichier appelé NOMS.

Sortez du programme en listant tous les noms ou en utilisant CTRL C. Tapez :

```
STORE A$,"NOMS"
ou STORE A$,"NOMS",S
```

cela dépend si vous sauvegardez à vitesse normale ou lente.

Mettez votre enregistreur de cassettes sur enregistrement et appuyez sur la touche RETURN.

Le message :

```
Saving... NOMS S
```

apparaîtra en haut de l'écran. Quand l'information contenue dans A$ aura été sauvegardée dans le fichier NOMS le message "Ready" apparaîtra. Rembobinez la bande au début du fichier.

L'information contenue dans le tableau A$ est maintenant stockée sur bande dans un fichier appelé NOMS. Cette information peut être chargée dans n'importe quel tableau de chaînes de caractères à deux dimensions à condition que ce tableau ait été dimensionné à des dimensions égales ou supérieures à celles de A$.

Nous allons maintenant voir comment récupérer l'information sur bande. Effacez la mémoire contenant le programme et entrez le programme suivant :

```
10 DIM B$(20,1)
20 CLS
30 PRINT"REMBOBINEZ JUSQU'AU DEBUT DU FICHIER"
40 PRINT "PUIS LISEZ LA BANDE"
```

La ligne 50 sera fonction de la vitesse à laquelle vous aurez sauvegardé votre programme (normale ou lente) et ce sera soit :

```
   50 RECALL B$,"NOMS"
ou 50 RECALL B$,"NOMS",S
```

Le reste du programme est :

```
60 PRINT CHR$(12)
70 INPUT"PREMIER CHIFFRE DU NUMERO DE TELEPHONE";D$
80 PRINT
90 FOR N=1 TO 20
100 IF B$(N,0)="EOF" THEN 70
110 IF LEFT$(B$(N,1,),1)=D$ THEN GOSUB 150
120 NEXT
130 END
140 REM SOUS-PROGRAMME
150 PRINT"NUMERO DE TELEPHONE",B$(N,1)
160 PRINT"NOM",,,B$(N,0)
170 PRINT
180 RETURN
```

Exécutez ce programme. La ligne 50 rappellera l'information stockée dans le fichier NOMS et la placera dans le tableau B$. Cette information est utilisée dans les lignes 60 à 180. Notez les messages qui apparaîtront en haut de l'écran quand vous rappellerez un fichier. Ils sont très similaires aux messages que vous obtenez quand vous chargez un programme.

B$ a les mêmes dimensions que A$ dans le programme précédent. Le fichier aurait pu aussi être chargé dans un tableau plus grand. Si par exemple, la ligne 10 avait été changée en :

```
10 DIM B$(30,1)
```

le programme aurait encore fonctionné. Les éléments B$ (21,0) à B$(30,0) et B$(21,1) à B$(30,1) n'auraient rien contenu et auraient pu être chargés à partir du programme si nécessaire. Vous pouvez donc garnir une partie du tableau à partir d'un fichier et le reste à partir d'un programme ou même par des commandes immédiates. Vous pouvez alors stocker le tableau résultant comme un autre fichier.

Vous pouvez aussi stocker des tableaux de nombres réels et de nombres entiers dans les fichiers. Par exemple, les programmes :

```
10 REM CREATION DE FICHIER
20 DIM C(12)
30 FOR N=0 TO 12
40 READ A
50 C(N)=A
60 NEXT
70 CLS
80 PRINT "COMMENCEZ L'ENREGISTREMENT"
90 WAIT 200
100 STORE C,"REEL",S:END
110 DATA 1,2,1,5,6,8,16,3,14,3,4,6,8,2,0,5
```

et

```
10 REM FILE RETRIEVAL
20 DIM D(14)
30 CLS
40 PRINT"REMBOBINEZ ET RELISEZ LA BANDE"
50 RECALL D, REEL,S
60 D(13)=42
70 D(14)=6.6
80 FOR N=0 TO 14
90 PRINT D(N),
100 NEXT
110 END
```

stockeront un tableau de réels de 13 éléments dans un fichier sur bande nommé REEL, rappelleront ce fichier et le garderont dans un tableau à 15 éléments, garniront les 2 éléments restant de ce tableau et imprimeront le contenu du nouveau tableau.

Ces deux programmes peuvent être modifiés pour fonctionner avec un tableau de nombres entiers. Vous pouvez faire cela à titre d'exercice.

Quand l'instruction RECALL est utilisée et qu'un fichier est chargé dans un tableau d'un seul élément, alors ce tableau doit être dimensionné de manière à avoir au moins 11 éléments. Autrement un message d'erreur "OUT OF MEMORY" apparaîtra.

Le message "ERROR FOUND" peut apparaître sur l'écran quand un fichier est rappelé. Cela ne veut pas nécessairement dire que les informations rappelées sont incorrectes, mais il est plus sage dans cette situation, soit de vérifier l'information soit de rappeler à nouveau le fichier.

GRAB

Dans un programme qui utilise beaucoup de mémoire pour les tableaux et les chaînes de caractères ou dans n'importe quel autre gros programme qui fonctionne en mode basse résolution, vous trouverez que GRAB est une instruction utile. GRAB vous permet d'utiliser l'espace mémoire qui serait mis de côté autrement pour le mode haute résolution. Cela vous permet d'avoir 7000 emplacements de mémoire supplémentaires. GRAB ne peut être utilisée si des périphériques (tel qu'un MODEM) sont utilisés car ils affectent "l'occupation" mémoire de la machine.

RELEASE

L'instruction RELEASE annule les effets de l'instruction GRAB et vous permet d'utiliser le mode haute résolution. Toutes les choses stockées dans l'espace mémoire rendu disponible par GRAB seront perdues si RELEASE est utilisée. Aussi GRAB et RELEASE doivent être utilisées avec précaution.

SAUVEGARDE D'AFFICHAGE A L'ECRAN

Vous pouvez sauver une partie ou un bloc de mémoire sur bande en utilisant la forme spéciale de l'instruction CSAVE. Celle-ci a le format :

```
CSAVE"...",Aaddr1,Eaddr2
```

pour la vitesse normale et

```
CSAVE"...",Aaddr1,Eaddr2,S
```

pour la vitesse lente.

Les nombres "addr1" et "addr2" sont les adresses du dernier emplacement de mémoire du bloc que vous souhaitez sauvegarder.

La construction d'un affichage à l'écran requiert parfois un gros programme et prend beaucoup de temps. Une des utilisations de ce cas spécial de l'instruction CSAVE est de sauvegarder sur bande le contenu des emplacements mémoire qui contrôlent l'affichage à l'écran. Quand cette information est rechargée dans l'ordinateur ultérieurement l'image sauvegardée apparaîtra de nouveau à l'écran. Il n'y a pas besoin de sauvegarder le programme qui a généré cet affichage.

Dans l'ORIC 16k l'écran basse résolution est contrôlé par des emplacements mémoire #3B80 à #3FE0 (15232 à 16352). Aussi pour sauvegarder un affichage sur un écran en mode basse résolution sur bande, pour la machine 16k nous pouvons utiliser :

```
CSAVE "SCREEN1", A#3B80,E#3FE0
```

ou si la vitesse lente est utilisée :

```
CSAVE "SCREEN1", A#3B80,E#3FE0,S
```

Pour recharger l'affichage dans la machine nous nous assurerons d'abord que la machine est en mode basse résolution et ensuite nous utiliserons :

```
CLOAD "SCREEN1",A#3B80,E#3FE0
ou CLOAD "SCREEN1",A#3B80,E#3FE0,S
```

selon la vitesse à laquelle l'information a été sauvegardée.

Dans l'ORIC 48k l'écran basse résolution est contrôlé par les emplacements mémoires #BB80 à #BFE0 (48000 à 49120).

L'écran en mode haute résolution dans la machine ORIC 16k est contrôlé par les emplacements mémoire #2000 à #3FE0 (8192 à 16352), et dans la machine 48k ORIC par les emplacements mémoire #A000 à #BFE0 (40960 à 49120).

Vous pouvez utiliser les nombres décimaux ou hexadécimaux avec ces commandes. S'il vous plaît assurez-vous que votre machine est dans un mode correct avant de charger l'affichage d'écran et chargez-le à la même vitesse que celle à laquelle il a été sauvegardé.

SAUVEGARDE D'ENSEMBLES DE CARACTERES

Si vous avez redéfini un certain nombre de caractères dans un programme, il est pratique de pouvoir sauvegarder cet ensemble de caractères et de le recharger en machine à chaque fois que vous souhaitez utiliser le programme. Cela vous évite d'avoir à redéfinir l'ensemble des caractères à chaque fois que le programme est utilisé.

La même technique est utilisée pour sauvegarder un ensemble de caractères comme pour sauvegarder un affichage à l'écran. Pour la machine ORIC 16k en mode basse résolution l'ensemble des caractères standard est contenu dans les emplacements mémoires #3400 à #3800 (13312 à 14336) et l'ensemble des caractères alternatifs est contenu dans les emplacements mémoire #3800 à #3B80 (14336 à 15232).

Les emplacements mémoire correspondants pour la machine 48k ORIC sont #B400 à #B800 (46080 à 47104) pour l'ensemble des caractères standard et #B800 à #BFE0 (47104 à 49120) pour l'ensemble des caractères alternatifs.

Il est plus inhabituel que l'ensemble des caractères soit sauvegardé et chargé en mode haute résolution. Si vous souhaitez le faire, pour la machine ORIC 16k, les emplacements mémoire nécessaires sont #1800 à #1C00 (6144 à 7186) pour l'ensemble des caractères standard et #1C00 à #2000 (7186 à 8192) pour l'ensemble des caractères alternatifs. Dans la machine ORIC 48k les emplacements mémoire correspondants sont #9800 à #9C00 (38912 à 39936) pour l'ensemble des caractères standard et #9C00 à #A000 (39936 à 40960) pour l'ensemble des caractères alternatifs.

TRUE (VRAI) et FALSE (FAUX)

TRUE et FALSE sont des nombres. TRUE vaut -1 et FALSE vaut 0. Leur but est de rendre certains programmes un peu plus facile à lire.

Par exemple :

```
100 IF POINT(X,Y)=TRUE THEN PRINT"PIXEL ENCRE"
```

et

```
10 REPEAT
---------
100 UNTIL FALSE
```

POP

Quand un sous-programme est appelé, l'ordinateur stocke l'adresse de l'instruction à laquelle le contrôle sera passé quand le sous-programme sera terminé dans un espace mémoire appelé le "haut de la pile".

Ceci est une explication un peu technique. Ce que cela signifie ? ORIC doit connaître toutes les adresses auxquelles il doit retourner quand il rencontre une instruction RETURN. Quand un sous-programme en appelle un autre, les adresses de retour des deux sous-programmes doivent être stockées dans la pile avec l'adresse du retour du sous-programme appelé le plus récemment sur le dessus de la pile.

Si vous utilisez l'instruction GOTO pour sortir du sous-programme, au lieu de sortir par l'instruction RETURN , alors l'adresse de retour pour ce sous-programme n'est pas enlevée du haut de la pile. Cela peut causer une erreur, la prochaine fois qu'une instruction RETURN est rencontrée.

Pour prévenir cela, nous utilisons l'instruction POP pour enlever toute adresse de retour non désirée du haut de la pile.

Si vous ne suivez pas tout à fait cette explication, ne vous en souciez pas. C'est un peu compliqué. Utilisez l'instruction POP à chaque fois que vous sortez d'un sous-programme par GOTO au lieu de quitter ce sous-programme par l'instruction RETURN. Cela peut être utilisé par exemple quand une erreur est supprimée à l'intérieur d'un sous-programme, ce qui fait que l'on saute directement à une routine de traitement d'erreurs standard. L'adresse du retour du sous-programme est enlevée par l'instruction POP.

En général on considère que c'est une bonne pratique de programmation de ne quitter les sous-programmes que par le biais de l'instruction RETURN, de manière que l'utilisation de POP soit réduite au minimum. Néanmoins, cette facilité est disponible si vous souhaitez l'utiliser.

POS

L'instruction POS fournit la position du curseur de texte sur l'écran. C'est un dispositif utile dans les programmes disons de traitement de texte, car si un mot est tapé à la fin de la ligne et est trop long, il est alors reporté sur la ligne suivante.

Dans la machine V1.0

A = POS

positionne la variable A (ou toute autre variable spécifiée) égale à un nombre compris entre 0 et 39 selon la position horizontale du curseur sur l'écran. 0 représente le bord gauche.

Dans la machine V1.1

A = POS(0)

est identique à A=POS(1) sur la machine V1.0 et

A=POS(1)

met dans A un nombre compris entre 0 et 255 selon la position de la plume de l'imprimante ORIC (voir chapitre suivant).

PULL

L'instruction PULL est plutôt similaire à POS. Elle permet de sauter hors d'une boucle REPEAT...UNTIL en utilisant une instruction GOTO et de revenir dans la boucle d'origine lorsque l'on rencontre une autre instruction UNTIL.

Faites exécuter le programme suivant :

```
10 TRON
20 REPEAT
30 : REPEAT
40 : GOTO 60
50 : UNTIL 0
60 F=F+1
70 PRINT F
80 UNTIL F=10
90 END
```

et notez l'ordre d'exécution des instructions. Ajoutez la ligne :

```
35 PULL
```

Faites fonctionner ce programme de nouveau et regardez comment PULL change cet ordre.

L'utilisation de PULL doit être réduite à un minimum. Comme POP elle doit être utilisée avant (ou après) un saut hors d'une boucle vers une routine de traitement d'erreurs.

ERREURS DANS LES TABLEAUX ET LES CHAINES DE CARACTERES

Si vous tapez de très longs programmes, ou si vous passez du mode TEXT au mode HAUTE RESOLUTION et inversement, il y a un risque que quelques-unes des chaînes de caractères du tableau soient corrompues. Cela est dû au fait que les zones de mémoire mises de côté pour les programmes BASIC se superposent aux zones utilisées pour stocker les chaînes de caractères et les tableaux.

Cette corruption peut être évitée grâce à l'instruction HIMEM qui modifie le montant de mémoire qui peut être occupé par le programme BASIC en repositionnant HIMEM à #97FF.

Utilisez cette instruction :

HIMEM #97 FF (48k) ou HIMEM #17FF (16k)

près du début de n'importe quel programme ou des chaînes de caractères ou des tableaux sont utilisés pour contenir des informations.

Nous rencontrerons l'instruction HIMEM de nouveau au prochain chapitre.

PERIPHERIQUES D'ENTREE ET DE SORTIE

Au moment où ce livre a été écrit, il y avait très peu de périphériques disponibles autres que les moniteurs, les téléviseurs, les enregistreurs de cassettes et les imprimantes, qui ont tous des prises spéciales ou portes. De manière à sélectionner les adresses en mémoire dans lesquelles on peut lire des données ou écrire des données, ORIC envoie des signaux électriques sur 16 fils connus sous le nom de lignes d'adresses. Une fois que l'emplacement mémoire est sélectionné, les données sont écrites ou lues en mettant des signaux électriques sur 8 fils connus sous le nom de lignes de données.

ORIC traite un périphérique d'entrée et de sortie à peu près de la même manière qu'il traite une zone de mémoire. Il adresse le périphérique et/ou bien lit l'information à partir de celui-ci, ou bien envoie l'information à celui-ci.

Certains périphériques d'entrée et de sortie nécessitent un traitement rapide. Ceci génère un signal appelé interruption qui fait que l'ordinateur arrête ce qu'il fait et traite le périphérique immédiatement.

Les 16 lignes d'adresses, les 8 lignes de données, la ligne de requête d'interruption (IRQ), la ligne de désactivation de la ROM (ROMDIS) sont disponibles à la sortie de la machine à la porte du bus d'expansion. Les autres signaux système disponibles à cette porte sont MAP, ϕ2, I/O (entrée-sortie), R/W (lecture, écriture), RESET et I/O contrôle (contrôle entrée et sortie). A moins que vous ne soyez un ingénieur électronicien, ou un spécialiste de systèmes informatiques, vous n'avez pas besoin de vous soucier de ces fonctions. Il vous suffit de savoir que les signaux suffisants sont déjà disponibles à cette porte pour vous permettre de connecter n'importe quel périphérique à cette machine. Certains périphériques tels que les contrôleurs de disque nécessiteront des extensions aux logiciels, mais la plupart des périphériques seront probablement contrôlés par des instructions PEEK et POKE.

TOUT SUR LES ATTRIBUTS

Tout au long de ce livre nous avons utilisé les attributs pour contrôler les coloris de l'arrière-plan et l'avant-plan sur les affichages à l'écran. Dans ce domaine, j'ai pris une démarche relativement pragmatique, en me concentrant plus sur ce qu'il fallait faire plutôt que sur le pourquoi cela était fait.

L'explication ci-dessous est pour ceux qui veulent connaître le pourquoi aussi bien que le comment. Cela n'est pas une lecture fondamentale. Vous pouvez très bien utiliser les attributs en suivant des procédures simples en utilisant les instructions PLOT, PRINT et POKE pour les affichages à l'écran sans rentrer en profondeur dans cette explication.

L'attribut est un nombre qui est contenu dans un bloc mémoire qui contrôle l'écran. Nous pouvons écrire des nombres de plusieurs manières. Ecrivons le nombre dans l'emplacement mémoire contrôlant l'écran comme un nombre binaire. Chaque emplacement mémoire contient 8 bits - B0 à B7.

B7	B6	B5	B4	B3	B2	B1	B0

Le nombre est identifié comme un attribut si les bits B5 et B6 sont tous deux à zéro.

Ainsi les nombres 0 à 31 (décimal) - 0000 0000 à 0001 1111 (en binaire) - sont identifiés comme des attributs quand ils sont stockés dans des positions mémoires qui contrôlent l'écran.

128 à 159 (en décimal) - 1000 0000 à 1001 1111 (en binaire) - sont aussi des nombres représentant des attributs. Nous y reviendrons bientôt.

Dans le mode basse résolution un seul emplacement mémoire contrôle une position de caractère. Nous avons vu au chapitre 12 comment le nombre dans la position de caractère (en faisant l'hypothèse qu'il est compris entre 32 et 127) fait que l'ordinateur cherchait dans d'autres emplacements mémoire la forme du caractère défini dans cette position, que ce soit un caractère standard, alternatif ou défini par l'utilisateur.

Toutefois, les nombres qui définissent la forme du caractère ne sont pas mis directement dans l'emplacement mémoire qui contrôle cette position sur l'écran. Ainsi quelque soit leur valeur ces nombres ne peuvent être des attributs. Un attribut doit être mis directement dans une position mémoire dédiée au contrôle de l'écran.

Ainsi dans les modes TEXT ou BASSE RESOLUTION de l'écran nous ne pouvons avoir qu'un attribut par position de caractère.

En mode basse résolution nous pouvons mettre un attribut directement dans la position mémoire nécessaire en utilisant POKE ou PLOT. Si nous utilisons l'instruction POKE nous devons calculer exactement quelle est la position mémoire qui contrôle la position du caractère à l'écran où nous souhaitons mettre l'attribut. En utilisant PLOT nous pouvons spécifier directement cette position de caractère en donnant les coordonnées X et Y sur l'écran.

Nous pouvons aussi mettre un attribut sur l'écran en utilisant :

```
PRINT CHR$(27)"...".
```

Ce qui se passe dans ce cas est que le caractère escape CHR$(27) modifie le code ASCII contenu dans l'emplacement mémoire en mettant le bit 6 de ce code ASCII à 0.

Prenons un exemple :

```
PRINT CHR$(27)"A"
```

Le code ASCII pour A est 65 (en décimal) ou #41. En notation binaire c'est

0100 0001

En mettant le bit B6 à 0 cela donne

0000 0001

qui est l'attribut dont le code est "1".

Ainsi : `PRINT @X,Y;CHR$(27)"A"`

est identique à :

```
PLOT X+1,Y,1
```

Nous savons maintenant qu'un attribut est identifié par des zéros dans les bits B5 et B6. L'effet produit par l'attribut est déterminé par les bits B0 à B4. Examinons ceci avec plus de détail :

F/B	CTL	B	G	R
B4	B3	B2	B1	B0
SCR		FL	DH	ALT

Considérons le bit B3 d'abord. C'est le bit de contrôle. Quand le bit B3 est à 0, le reste des bits est assez facile à comprendre. Ils contrôlent la couleur.

B3=0

B0 contrôle le ROUGE
B1 contrôle le VERT
B2 contrôle le BLEU
B4 permet de basculer entre l'arrière-plan et l'avant-plan.

Toutes les couleurs sur votre poste de télévision sont faites de combinaisons de lumières rouge, verte et bleue. Le mélange de rayons lumineux n'est pas tout à fait identique aux mélanges de peinture. Par exemple, le mélange des lumières rouge et verte donne une lumière jaune.

Ainsi, si le bit B4=0 (avant-plan) B3=0, B2=0 (pas de lumière bleue), B1=1 (lumière verte), B0=1 (lumière rouge), nous avons un attribut d'avant-plan de couleur jaune. En positionnant le bit 4 à 1 nous obtiendrons un attribut de couleur jaune pour l'arrière-plan. En convertissant cela en décimal cela donne le code 3 pour la couleur jaune pour l'avant-plan et le code 19 pour la couleur jaune pour l'arrière-plan.

Quand le bit B3=1 les choses sont un peu plus compliquées. Nous devons considérer également le bit B4 comme un bit de contrôle.

B3=1, B4=0

B0=0 pour les caractères standard et =1 pour les caractères alternatifs.

B1=0 pour la simple hauteur et =1 pour les caractères double hauteur.

B2=0 pour les caractères clignotants et =1 pour les caractères permanents.

Dans ce cas B0 et B1 sont opérationnels seulement sur l'écran en mode basse résolution.

Par exemple, B4=0, B3=1, B2=1, B1=0 et B0=1 donnent des caractères alternatifs de simple hauteur clignotants. La conversion en décimal donne le code 13 pour cet attribut.

B3=1, B4=1

Ces attributs doivent être évités, spécialement si B1=0 (attribut relatif au 60 Hz). Ils affectent la synchronisation de l'écran et leurs effets sont imprévisibles.

Un attribut d'avant-plan ou un caractère alternatif double hauteur clignotant affecte la rangée de positions de caractères qui le suit, mais met un espace vide dans la position de caractère sur l'écran qui correspond à l'emplacement mémoire contenant cet attribut.

Un attribut d'arrière-plan non seulement affecte les positions de caractères qui le suivent dans la ligne mais produit aussi un changement de la couleur d'arrière-plan, dans la position du caractère qui correspond à l'emplacement mémoire qui contient cet attribut.

Ainsi, quand nous spécifions les couleurs d'arrière-plan et d'avant-plan dans une ligne d'affichage à l'écran, habituellement nous spécifions d'abord l'attribut d'arrière-plan puis l'attribut d'avant-plan. C'est le cas à la fois dans les modes haute et basse résolutions.

Finalement, revenons au bit B7. C'est le bit inverse. Il inverse les signaux rouge, vert et bleu contrôlés par les bits B0 à B2 quand le bit B3 est positionné à 0.

Prenons par exemple, l'attribut couleur jaune d'avant-plan (code 3) qui a les lumières rouge et verte positionnées et la lumière bleue non positionnée. Si nous mettons le bit B7 à 1, le code devient 131 et les couleurs sont inversées. Le rouge et le vert sont dépositionnés et le bleu est positionné. Ainsi l'attribut de couleur jaune d'avant-plan devient un attribut de couleur bleue d'avant-plan.

Un grand nombre de choses peuvent être faites en utilisant les attributs, suffisamment d'ailleurs pour écrire un autre livre! Les effets spectaculaires peuvent être créés en imprimant des formes et des messages à l'écran en utilisant la même couleur d'encre et de papier, de telle manière qu'elles soient invisibles initialement et visibles ensuite en changeant l'attribut sur l'écran. Comme les attributs sont des nombres, ils peuvent être stockés en mémoire avec d'autres nombres définissant une forme et tout le bloc de mémoire (connu sous le nom de **macro graphique**) transféré à des emplacements contrôlant l'écran haute résolution. En changeant soudainement les attributs d'arrières-plan, en utilisant à nouveau le mode haute résolution cela peut faire que des "rayons de la mort" tirent très rapidement à travers l'écran dans des programmes de jeux.

Il n'y a pas suffisamment de place dans un livre d'introduction pour étudier cela en détail. J'espère que vous explorerez ce qui peut être fait avec les attributs vous-mêmes et que mes suggestions vous seront de quelque aide.

Dans le mode haute résolution, chaque position de caractère est contrôlée par 8 positions mémoires séparées. Les pixels sur l'écran sont directement contrôlés par les bits B5 à B0 des nombres qui se trouvent dans les emplacements mémoire.

Vous pouvez vous souvenir que quand on utilise l'instruction FILL pour encrer des pixels sur l'écran, le premier pixel doit toujours être encré. En d'autres termes, B5 doit être positionné à 1. La raison en est qu'autrement le modèle est interprété comme un attribut si l'instruction FILL commence au début d'une position de caractère. Souvenez-vous que la condition pour être attribut est que B5 et B6 soient à 0.

Quand vous mettez un caractère sur l'écran haute résolution en utilisant l'instruction CHAR, alors les nombres définissant le caractère sont mis dans la mémoire qui contrôle l'écran. Ceci est dû au fait qu'en mode haute résolution vous avez une correspondance directe 1 pixel = 1 bit. Pourquoi ces nombres, dont certains sont inférieurs à 32 (décimal), ne sont-ils pas traités comme des attributs ? La réponse est que l'instruction CHAR positionne **toujours** à 1 le bit B6 de tout nombre qu'elle met dans les emplacements mémoire contrôlant l'écran. Ainsi vous pouvez utiliser CHAR tranquillement pour le but qui lui est assigné (intelligent non !)

Chapitre 15

COMMENCEZ A IMPRIMER

L'IMPRIMANTE D'ORIC

L'imprimante d'ORIC utilise quatre stylos bille minuscules pour écrire ou dessiner sur un rouleau de papier ordinaire. Vous pouvez sélectionner, sous le contrôle du programme, des encres de couleur noire, bleue, rouge ou verte.

La vitesse d'impression est de 12 caractères par seconde. La vitesse du tracé de dessin est de 52 mm par seconde dans la direction horizontale (l'axe des x) et 73 mm par seconde dans la direction verticale (l'axe des y).

POSITION DE LA PLUME

Les positions normales de la plume sont :

0 noire
1 bleue
2 verte
3 rouge

Vous pouvez, si vous le désirez, changer la couleur d'une plume de n'importe laquelle de ces quatre positions. Je ferai ici l'hypothèse que les plumes sont normales.

MODE D'IMPRESSION

Il y a deux modes d'impression qui peuvent être sélectionnés par le programme. Ce sont le mode texte et le mode graphique.

MODE TEXTE

Quand vous mettez ORIC sous tension, l'imprimante est en mode texte. La plume est dans le coin supérieur gauche du papier et la position de la plume 0 (noire) est sélectionnée.

CARACTERES PAR LIGNE

En mode texte vous pouvez imprimer normalement 80 caractères par ligne. Pour sélectionner la largeur de ligne de 40 caractères, utilisez l'instruction :

POKE 49,53 (V1.0) ou POKE 598,40 (V1.1).

Pour revenir à 80 caractères par ligne utilisez :

POKE 49,93 (V1.0) ou POKE 598,80 (V1.1).

LLIST

LLIST liste le programme sur l'imprimante de la même manière que LIST liste le programme sur l'écran.

LPRINT

LPRINT imprime des messages sur l'imprimante. Son fonctionnement est très similaire à PRINT.

Essayez :

```
LPRINT 26
LPRINT"SALUT"
LPRINT CHR$(65)
```

LPRINT peut être utilisé avec TAB, les séparateurs virgule et apostrophe, de la même manière qu'avec PRINT. LPRINT @ n'est toutefois pas permis.

CODES DE CONTROLE

LPRINT peut être utilisé avec différents codes de contrôle. Je vais d'abord les lister et ensuite nous les examinerons plus en détail.

```
LPRINT CHR$(8)  - recule d'une position
LPRINT CHR$(11) - inverse le saut de ligne
LPRINT CHR$(17) - sélectionne le mode texte
LPRINT CHR$(18) - sélectionne le mode graphique
LPRINT CHR$(29) - change la position de la plume par
                  rotation.
```

CHR$(8)

LPRINT CHR$(8) fait que la tête d'impression se déplace d'une position vers l'arrière (vers la gauche). C'est très utile pour créer des caractères "composites". Par exemple :

```
10 REM SEPT EUROPEEN
20 LPRINT 7;
30 LPRINT CHR$(8);
40 LPRINT CHR$(45)'MOINS
```

CHR$(11)

LPRINT CHR$(11) déplace la tête d'impression d'une ligne vers le haut. (En fait il redescend le papier d'une ligne). C'est très utile pour créer des caractères gras par surimpression. Essayez :

```
10 A$="CARACTERE GRAS"
20 LPRINT A$
30 LPRINT CHR$(11);
40 LPRINT A$
```

CHR$(17)

LPRINT CHR$(17) met l'imprimante en mode texte. Cette instruction n'a pas d'effet sur l'imprimante qui est déjà en mode texte.

CHR$(18)

LPRINT CHR$(18) met l'imprimante en mode graphique. Nous étudierons cela avec plus de détails bientôt.

CHR$(29)

LPRINT CHR$(29) change la position de la plume par rotation. Si la position de la plume est à 0 (noire), alors LPRINT CHR$(29) change la position en 1 (bleue). La prochaine instruction LPRINT CHR$(29) changera la position en 2 (verte) et ainsi de suite. Essayez :

```
10 FOR N=0 TO 4
20 READ A$
30 LPRINT A$
40 LPRINT CHR$(29);
50 NEXT
60 DATA NOIR,BLEU,ROUGE,VERT,ET NOIR DE NOUVEAU.
```

Note - Les instructions LLIST et LPRINT CHR$(8), CHR$(11), et CHR$(29) ne fonctionnent qu'en mode texte.

MODE GRAPHIQUE

L'instruction LPRINT CHR$(18) met l'imprimante en mode graphique. Dans ce mode la "carte" de l'imprimante est divisée en pas ayant chacun 0,2 mm de long. Il y a 480 de ces pas dans la direction horizontale et un nombre infini (en théorie) dans la direction verticale (vers le haut et vers le bas). Pour des raisons pratiques nous limiterons le mouvement vertical entre -999 (999 pas vers le bas du papier) et +999 (999 pas vers le haut du papier) à partir de l'origine. Si nous voulons étendre cet intervalle nous pouvons repositionner l'origine.

Quand l'imprimante est mise en mode graphique, l'origine est positionnée sur le côté gauche et la position 0 de la plume est sélectionnée. Ainsi, si nous dessinons une ligne de 480 pas dans la direction horizontale (x=+480), nous obtiendrons une ligne horizontale en travers de la zone de dessin.

COMMANDES IMPLICITES

Pour commander l'imprimante en mode graphique, ORIC utilise ce qui est connu sous le nom de "commandes implicites". Elles sont toutes basées sur l'instruction LPRINT qui doit être suivie par une chaîne de caractères. Le premier caractère dans la chaîne détermine ce que la commande fait. Ce premier caractère n'est pas lui-même imprimé.

Comparez cela avec l'instruction PRINT CHR$(27) (voir

chapitre 7), où le premier caractère de la chaîne détermine le format d'affichage.

Comme d'habitude, je listerai d'abord ces commandes et ensuite je les expliquerai avec plus de détails. Utilisez la touche manuelle d'alimentation du papier sur l'imprimante de telle manière que vous ayez du papier vierge pour chaque démonstration.

LPRINT"A" - retourne au mode texte.
LPRINT"Cn" - sélectionne la couleur de l'encre (n vaut 0,1,2 ou 3).
LPRINT"D" - définit la position courante au point absolu spécifié.
LPRINT"H" - retourne à l'origine.
LPRINT"I" - positionne l'origine.
LPRINT"J" - tire une ligne de la position courante de la plume à une position relative spécifiée.
LPRINT"In" - spécifie le titre de la ligne (n est compris entre 0 et 15).
LPRINT"M" - déplace la plume vers un point absolu spécifique sans tirer de ligne.
LPRINT"P" - imprime des caractères sans revenir au mode texte.
LPRINT"Sn" - spécifie la taille des caractères (n est compris entre 0 et 63).
LPRINT"Qn" - spécifie la direction de l'imprimante (n vaut 0, 1, 2, ou 3).
LPRINT"R" - déplace la plume vers une position relative spécifiée sans tirer de ligne.
LPRINT"X" - trace un axe à partir de la position actuelle de la plume et marque à des intervalles spécifiés.

LPRINT"A"

Cette instruction déplace la plume vers la gauche de la zone de dessin sans tirer de ligne et met l'imprimante à nouveau en mode texte. LPRINT CHR$(17) peut aussi être utilisée pour mettre l'imprimante en mode texte.

LPRINT"Cn"

Quand l'imprimante est mise en mode graphique, la plume est en position 0 (noire). Si aucune couleur n'est spécifiée des lignes noires seront tirées. LPRINT"Cn" spécifie la couleur :

n = 0(noire) n = 1(bleue) n = 2(verte) n = 3(rouge).

La couleur choisie restera constante jusqu'à ce qu'une autre couleur soit sélectionnée.

Essayez :

```
10 LPRINT CHR$(18)
20 LPRINT"C3"
30 LPRINT"PLES ROSES SONT ROUGES"
40 LPRINT"C1"
50 LPRINT"PLES VIOLETTES SONT BLEUES"
60 LPRINT"C2"
70 LPRINT"PL'HERBE EST VERTE"
80 LPRINT"C0"
90 LPRINT"PQUEL PROGRAMME ENNUYEUX"
100 LPRINT"A"
110 REM RETOUR AU MODE TEXTE.
```

LPRINT"D"

LPRINT"D,X,Y" tire une ligne de la position courante de la plume au point de coordonnées X,Y, où X est le nombre de pas à partir de l'origine dans la direction X (horizontale) et Y est le nombre de pas à partir de l'origine dans la direction Y (verticale). Les coordonnées absolues ont été expliquées au chapitre 13.

X est compris entre l'intervalle -480 à +480 (selon la position d'origine) et Y est compris dans l'intervalle de -999 à +999.

Un certain nombre de points peuvent être spécifiés les uns après les autres. Essayez :

```
10 LPRINT CHR$(18)
20 LPRINT "D0,300,300,300,300,0,0,0"
30 LPRINT"A"
```

Ceci tire une ligne depuis l'origine (0,0) au point (0,300), de là au point (300,300), et de là à (300,0) et enfin retour à l'origine (0,0). Ainsi un carré de côté dont la longueur correspond à 300 pas est dessiné.

LPRINT"H"

Ceci ramène la plume à l'origine sans dessiner de ligne. Essayez :

```
10 LPRINT CHR$(18)
20 LPRINT"C3"
30 FOR Y=0 TO 300 STEP 30
40 A$="D300,"+STR$(Y)
50 FRE(""):REM PAS D'ESPACE
60 LPRINT A$
70 LPRINT"H"
80 NEXT
90 LPRINT"A"
```

LPRINT"I"

Ceci change l'origine de la position courante de la plume. Essayez :

```
10 LPRINT CHR$(18)
20 FOR N=0 TO 2
30 GOSUB 100
40 LPRINT"I"
50 NEXT
60 LPRINT"A"
100 LPRINT"D0,100,100,100,100,0,0,0,100,100"
120 RETURN
```

LPRINT"J"

Cette instruction fonctionne comme LPRINT"D" excepté que X et Y expriment les coordonnées des déplacements relatifs à partir de la position courante de la plume. Les coordonnées relatives ont été expliquées au chapitre 13. Essayez :

```
10 LPRINT CHR$(18)
20 LPRINT"J0,300,300,0,0,-300,-300,0"
30 LPRINT"A"
```

LPRINT"Ln"

Cette instruction spécifie le titre de la ligne à dessiner. Des lignes continues sont dessinées par défaut. Il y a 16 types de lignes y compris les lignes continues :

n	Type de Ligne
0	————————————————
1	.....
2	.....
3	----------
4	----------
5	----------
6	----------
7	----------
8	----------
9	----------
10	----------
11	— — — — — — — — — — — — — —
12	— — — — — — — — — — — — — —
13	— — — — — — — — — — — — — —
14	— — — — — — — — — — — — — —
15	— — — — — — — — — — — — — —

Essayez d'ajouter :

15 LPRINT"L6"

au programme précédent.

LPRINT"M"

Cette instruction fonctionne de la même manière que LPRINT"D", excepté qu'aucune ligne n'est dessinée quand la plume se déplace.

LPRINT"P"

Cela permet à des caractères d'être imprimés alors que l'imprimante reste en mode graphique. Les caractères doivent être dans une chaîne de caractères. Nous avons vu l'utilisation de LPRINT"P" un peu plus tôt quand nous avons étudié LPRINT"Cn".

LPRINT"Sn"

Cela vous permet de sélectionner la taille du caractère. Vous pouvez avoir 64 tailles différentes de caractères ; n prend les valeurs comprises entre 0 (la plus petite) et 63 (la plus grande).

Le nombre de caractères que vous pouvez avoir sur une ligne dépend de la taille du caractère. Les caractères les plus petits (n=0) seront imprimés à raison de 80 par ligne. Les caractères les plus grands (n=63) requièrent une ligne entière de caractères.

Le programme suivant calculera combien de caractères d'une taille spécifique peuvent être mis sur une ligne.

```
10 INPUT"TAILLE DU CARACTERE(0-63)";TAILLE
20 NUM=INT(80/(TAILLE+1))
30 PRINT"LA TAILLE"TAILLE"DONNE"NUM"CARACTERES PAR LIGNE"
```

Les spécifications de l'imprimante garantissent la précision de cette formule seulement pour les tailles comprises entre 0 et 15 (80 à 5 caractères par ligne). J'ai trouvé que cela donnerait un guide utile pour cet intervalle de taille de caractère.

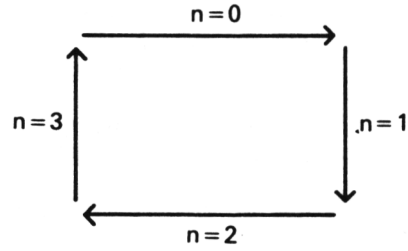
Essayez :

```
10 LPRINT CHR$(18)
20 LPRINT"C3"
30 LPRINT"S38"
40 LPRINT"PAN"
50 LPRINT"C2"
60 LPRINT"S18"
70 LPRINT"PORIC"
80 LPRINT"C0"
90 LPRINT"S0"
100 A$="EXTRAVAGANCE"
110 B$=A$+A$+A$+A$+A$+A$:FRE("")
120 LPRINT"P"B$
130 LPRINT"A"
```

Si aucune taille de caractère n'est spécifiée, S0 est pris par défaut.

LPRINT"Qn"

Normalement ORIC imprime les caractères de gauche et droite. LPRINT"Qn", où n peut valoir 0,1,2 ou 3, vous permet de spécifier une autre direction, c'est-à-dire du haut vers le bas, du bas vers le haut, et de la droite vers la gauche.



Essayez :

```
10 LPRINT CHR$(18)
20 LPRINT"S7"
30 LPRINT"M96,0"
40 A$="CARRE"
50 FOR N=0 TO 3
60 LPRINT"Q"STR$(N)
70 LPRINT"P"A$
80 NEXT
90 LPRINT"S0":LPRINT"C0":LPRINT"A"
```

LPRINT"R"

Cette instruction fonctionne de la même manière que LPRINT"J", sauf qu'aucune ligne n'est tracée quand la plume se déplace.

LPRINT"X,A,S,I"

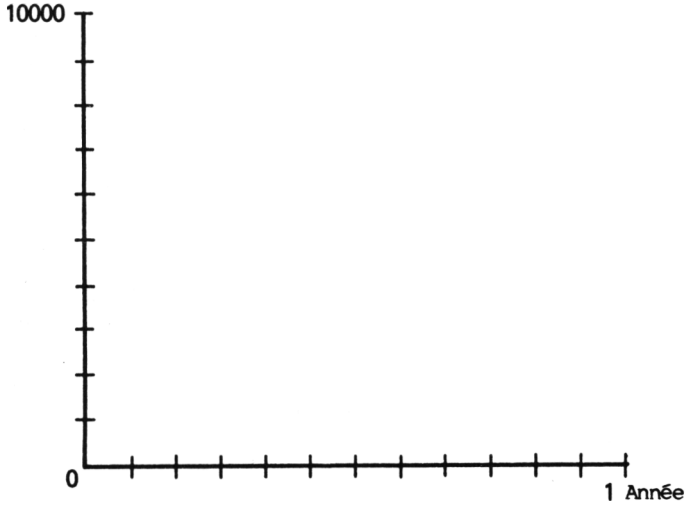
Ceci est utilisé pour dessiner des axes pour les graphiques. Ceci est un peu mathématique, si les maths vous effraient, ignorez cette instruction.

A, S et I sont des nombres. A vaut soit 0 soit 1 et sélectionne l'axe à dessiner :

A=0 donne un axe vertical (axe de Y)
A=1 donne un axe horizontal (axe de X)

Les axes sont normalement gradués en section par des petits tirets appelés des grades. Par exemple, supposez que l'axe des X représente le temps, disons une année. Cet axe pourra être marqué de 12 sections égales, chacune représentant un mois.

Supposez que l'axe des Y représente le total des ventes d'un produit. La longueur totale de l'axe pourrait représenter 100000 ventes et cela pourrait être décomposé en 10 sections égales, chacune représentant 10000 ventes.



S représente la taille en pas de 2 mm de chaque section ou graduation sur l'axe. C'est un nombre compris entre -999 et +999. Si S est négatif, l'axe est dessiné vers le bas (axe des Y) ou vers la gauche (axe des X).

I spécifie le nombre de sections et est compris entre 1 et 255.

Essayez :

```
10 LPRINT CHR$(18)
20 LPRINT"M240,-270"
30 LPRINT"I"
40 LPRINT"X1,30,7"
50 LPRINT"H"
60 LPRINT"X1,-30,7"
70 LPRINT"H"
80 LPRINT"X0,25,10"
90 LPRINT"H"
100 LPRINT"X0,-25,10"
110 LPRINT"A"
```

AUTRES IMPRIMANTES

ORIC pilote les imprimantes par ce qui est appelé une porte parallèle ou sortie "Centronics". Il peut fonctionner avec toute imprimante qui a une entrée Centronics à condition que vous puissiez obtenir un câble de connection adéquat. Bien que l'interface Centronics utilise des signaux standard, ces signaux peuvent être amenés sur différentes broches, sur différentes imprimantes.

Vous n'avez pas à vous soucier de cela si vous utilisez l'imprimante ORIC car cette imprimante a un câble de connection spécialement conçu pour la sortie ORIC.

CONNECTION DES BROCHES

Si vous avez une autre imprimante vous pouvez acheter un câble de connection adéquat auprès d'un revendeur d'ordinateur. Sinon vous devrez en faire un, ou trouver un expert en électronique qui vous le fera.

Les signaux de la sortie ORIC sont :

BROCHE	SIGNAL
1	STB
3	D0
5	D1
7	D2
9	D3
11	D4
13	D5
15	D6
17	D7
19	ACK

Toutes les broches paires sont connectées à la terre.

Les numéros des broches et les signaux d'entrée correspondants sont donnés dans la documentation fournie avec toute imprimante de type parallèle.

Par exemple, les connections pour l'imprimante Microline 80 sont :

BROCHE	SIGNAL
1	STB
2	D0
3	D1
4	D2
5	D3
6	D4
7	D5
8	D6
9	D7
10	ACK

Les broches 19 à 30 sont connectées à la terre.

IMPRIMANTE COULEUR TRS-80

L'imprimante graphique couleur TRS-80 CGP-115 de Radio Shack est très similaire à l'imprimante ORIC. En faisant l'hypothèse que vous avez le câble de connection adéquat tout programme que vous écrirez pour piloter l'imprimante ORIC fonctionnera aussi avec ce périphérique.

POS(1)

Pour la machine V1.1 POS(1) fournit la position horizontale du curseur d'impression (voir chapitre 14).

Chapitre 16

CODE MACHINE

INSTRUCTIONS MACHINE

Les ordinateurs fonctionnent uniquement avec des 0 et des 1. Quand vous dites à ORIC de faire quelque chose, en utilisant des mots qui sont compréhensibles par les humains (PRINT, DRAW, ZAP), ces instructions sont traduites en chaîne de nombres binaires qu'ORIC peut comprendre. Il y a un ensemble de programmes constamment présents dans la mémoire de l'ordinateur qui font cela. Ce groupe de programmes est connu sous le nom de **SYSTEME D'EXPLOITATION** (en anglais **OPERATING SYSTEM**).

Les nombres binaires que l'ordinateur comprend sont appelés "code machine" (ou instructions machine). Si vous connaissez ce que chaque nombre fait, vous pouvez communiquer directement avec la machine dans son propre langage.

La notation hexadécimale est une manière pratique d'écrire en abrégé les nombres binaires. Quand nous tapons des instructions machines directement dans la machine, nous utilisons normalement le format hexadécimal. Pour nous permettre de nous souvenir de ce que chaque nombre fait nous lui donnons un nom : mnémonique.

Ainsi par exemple :

1110 1000 (=#E8) a pour mnémonique INX.

Un programme qui permet de transformer ces mnémoniques en langage machine est appelé un assembleur. Un programme qui change un langage de haut niveau (exprimé en général en anglais) tel le BASIC en code machine est appelé un compilateur ou un interpréteur.

Les programmes contenus dans le système d'exploitation de l'ordinateur pour permettre ces dernières fonctions sont généralement interprétés. Un compilateur convertit un programme BASIC en code machine avant que le programme soit chargé dans l'ordinateur.

Programmer en code machine n'est pas difficile, bien que quelques programmeurs en code machine prétendent le contraire. Surtout c'est terriblement ennuyeux. Les programmeurs en code machine doivent connaître un peu plus de choses relatives à la structure interne de la machine.

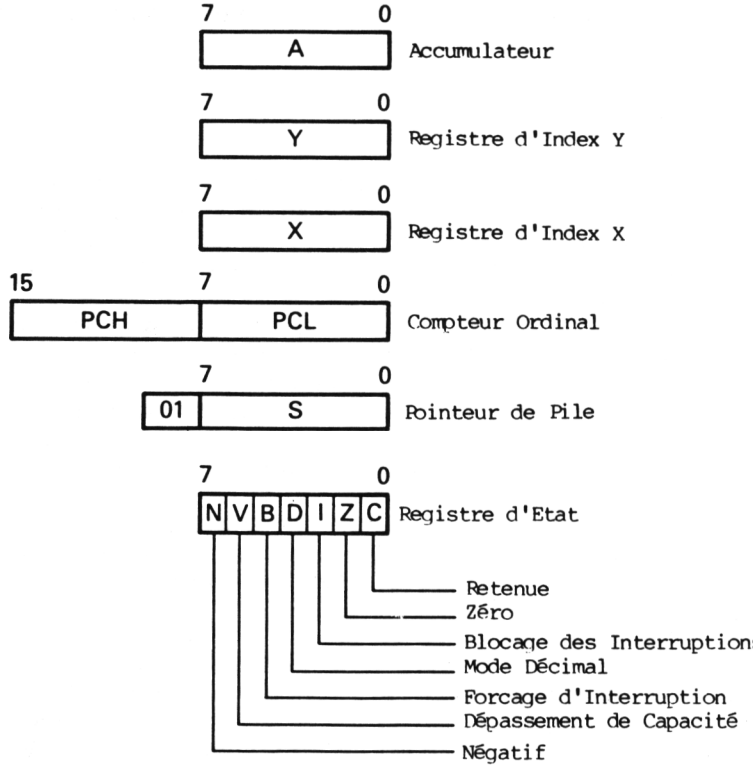
LE MICROPROCESSEUR 6502

ORIC est contrôlé par le microprocesseur 6502. Le microprocesseur fonctionne tout le temps que la machine est sous tension. Il lit les nombres binaires à partir de la mémoire, interprète les nombres comme des instructions, les données ou des adresses mémoire, et il exécute les opérations sur la base de ces informations.

Pour ce faire le microprocesseur utilise des emplacements spéciaux connus sous le nom de registres. Ces registres ne font pas partie de la mémoire principale de la machine et sont identifiés par leur nom plutôt que par une adresse numérique.

Ces registres sont :

L'accumulateur (A)
Les index de registres (X et Y)
Le compteur ordinal (PC)
Le pointeur de pile (S)
Le registre d'état (P).



L'accumulateur est surtout concerné par la plupart des opérations arithmétiques et logiques faites par la machine. Les nombres résultant de ces opérations sont d'habitude stockés dans l'accumulateur.

Les registres d'index sont utilisés pour stocker des valeurs pour compter, pour mesurer l'écoulement du temps et pour "indexer" (identifier une adresse mémoire ou une série d'adresses mémoire par référence à une adresse de base).

Le compteur ordinal contient l'adresse mémoire de l'instruction qui est présentement exécutée.

Le pointeur de pile contient l'adresse du début de section de la mémoire connue comme la pile. La pile est utilisée pour contenir l'information quand le microprocesseur est interrompu dans sa tâche principale et doit faire quelque chose d'autre. Par exemple les adresses de retour de sous-programme sont stockées dans la pile.

Le registre d'état est une collection de bits ou "drapeaux" qui peuvent être positionnés à 1 ou repositionnés à 0 selon le résultat des opérations dans la machine. Par exemple le bit B0 du registre d'état est le bit de retenue. Si une opération arithmétique effectuée dans le microprocesseur a un résultat supérieur à 255 ce drapeau doit être positionné à 1.

LANGAGE ASSEMBLEUR

Regardons de nouveau les noms ou "mnémoniques" que nous donnons aux instructions machine.

Par exemple :

1110 1000 a pour mnémonique INX.

Cette instruction fait que le nombre contenu dans le registre d'index X est augmenté de 1. C'est-à-dire que le nombre contenu dans X après que l'instruction ait été exécutée est égal au nombre qui était contenu précédemment dans X plus un.

INX est une abréviation pour INcrémenter X.

Le nombre 1110 1000 est connu sous le nom de code opération ou code-opératoire pour l'instruction INX. Les codes opérations sont généralement écrits en hexadécimal.

Ainsi :

#E8 est le code opération pour INX.

En général, les mnémoniques utilisés pour représenter des nombres que le microprocesseur interprète comme des instructions réfèrent aux divers registres et drapeaux. Par exemple :

STA signifie STore the Accumulator in memory (stocker l'accumulateur en mémoire).

LDA signifie LoaD the Accumulator (charger l'accumulateur).

SEC signifie SEt the Carry Flag (positionner le drapeau de retenue).

Le groupe de mnémonique ainsi généré est connu sous le nom de **Langage Assembleur** pour le microprocesseur 6502. Une liste des instructions du langage assembleur pour le 6502 est donné en annexe F.

Les "programmeurs en langage machine" ne programment pas réellement en code machine. Ils écrivent leur programme en langage assembleur et le convertissent en code machine ensuite. Le programme en langage assembleur est connu sous le nom de "programme source". Le programme en langage machine résultant est connu sous le nom de "programme objet".

Tous les nombres contenus dans le code machine ne sont pas des codes opérations. Considérez, par exemple, les instructions :

LDA #21

Cela signifie charger l'accumulateur avec la valeur hexadécimale 21.

Note - En langage assembleur tous les nombres sont hexadécimaux. Le symbole # ne signifie pas ici hexadécimal. Il signifie que l'accumulateur est chargé directement avec un nombre plutôt qu'avec le contenu d'un autre registre ou d'un emplacement mémoire.

Le code machine pour LDA #21 est :

#A9 #21

#A9 est le code opération pour LDA#, mais #21 est simplement un nombre qui est chargé dans A.

Un autre exemple est :

STA 0402

Ce qui signifie STocker l'Accumulateur dans une position mémoire dont l'adresse est 0402 (en hexadécimal), c'est-à-dire changer le nombre contenu dans cet emplacement mémoire de telle manière qu'il soit égal au nombre contenu dans l'accumulateur.

Le code machine pour ceci est :

#8D #02 #04

#8D est le code opération pour STA.

#02 et #04 donnent l'adresse à laquelle l'accumulateur doit être stocké. Les adresses en code machine sont écrites en spécifiant les unités d'abord !!

Les nombres dans les instructions machines peuvent donc être des codes opérations, des données ou des adresses. Le microprocesseur les interprète selon ce qui précède.

CALL

CALL permet d'appeler une routine écrite en langage machine à partir du BASIC. CALL est une instruction BASIC. Elle adresse l'emplacement mémoire où la première instruction de la routine en langage machine est stockée. Si par exemple une routine en code machine commence à l'adresse #400 alors la commande :

```
CALL #400
```

transférera le contrôle à la routine en code machine.

RTS

RTS est une instruction en langage assembleur. RTS signifie ReTurn from Subroutine (retour d'une routine) et son code opération est #60. Si la routine en code machine est appelée à partir d'un programme BASIC, RTS contrôlera le retour à l'instruction suivante de ce programme.

Si CALL est utilisé comme une commande directe, RTS retourne à la machine en mode direct.

RTS est normalement la dernière instruction de toute routine écrite en code machine.

COMMENT METTRE DES CODES MACHINES EN MEMOIRE

Le code machine est une série de nombres. Ceux-ci peuvent être placés dans les positions mémoires en utilisant l'instruction POKE directement ou dans un programme BASIC.

Ceci est mieux illustré par un exemple. La routine suivante place le nombre #41 dans la position mémoire #0415. La routine doit commencer dans la position mémoire #0400.

```
LDA #41
STA 0415
RTS
```

L'assemblage en code machine donne :

```
#A9 #21
#8D #15 #04
#60
```

Nous pourrions utiliser l'instruction POKE #A9 dans l'emplacement mémoire #0400, #21 dans l'emplacement mémoire #0401 et ainsi de suite. Ou encore utiliser :

```
10 FIRST=#400
20 NUM=6
30 FOR K=0 TO NUM
40 READ X
50 POKE FIRST+K,X
60 NEXT:END
70 DATA #A9, #41:REM LDA #41
80 DATA #8D,#15,#04:REM STA 0415
90 DATA #60:REM RTS
```

Faites exécuter ce programme. Il placera la routine en code machine en mémoire, mais elle ne l'exécutera pas. Si vous examinez l'emplacement mémoire #0415, il ne doit pas contenir #41.

Si toutefois, vous entrez la commande :

```
CALL #0400
```

l'ordinateur exécutera la routine en code machine et reviendra à l'état "Ready". L'emplacement mémoire #0415 contiendra maintenant #41.

HIMEM

Où les routines en instruction machine peuvent-elles être placées en mémoire ? La réponse est qu'en théorie elles peuvent être mises n'importe où dans la mémoire RAM réservée à l'utilisateur. En pratique, toutefois, nous devons faire attention à ce que le code machine ne soit pas écrasé par la lecture du programme BASIC.

Il y a une zone entre les emplacements mémoires #0400 et #0420 qui est réservée pour les instructions en code machine. Mais, ceci ne donne que 32 emplacements. Nous devons trouver un autre endroit pour des programmes en code machine.

Nous pourrions mettre des programmes en code machine dans le haut de la mémoire de l'utilisateur. Nous avons vu dans le chapitre 14 que nous pouvons changer l'adresse mémoire maximum que les programmes BASIC peuvent utiliser à l'aide de l'instruction HIMEM.

Normalement cette adresse marque le haut de la mémoire utilisateur, #97FF pour la machine 48k et #17FF pour la machine 16k. Si toutefois nous changeons HIMEM pour une valeur plus basse, nous pouvons empêcher le programme BASIC d'utiliser le haut de la mémoire d'utilisateur créant ainsi un emplacement "sûr" pour les instructions machine.

Supposons que nous souhaitions réserver 240 emplacements mémoire pour contenir du code machine - arrondissons cela à 255 emplacements de manière à avoir quelques uns de reste. 255 vaut #FF en hexadécimal.

Ainsi :

HIMEM #9700 (pour la machine 48k)
ou HIMEM #1700 (pour la machine 16k)

empêchera le programme BASIC d'utiliser les 255 premiers emplacements de la mémoire utilisateur. Nous pouvons alors utiliser l'instruction POKE pour mettre des routines en code machine commençant en #9701 (pour la machine 48k) ou #1701 (pour la machine 16k).

DEEK ET DOKE

Nous avons vu que, quand nous mettons des instructions machine à l'aide de l'instruction POKE, nous stockons des nombres plus grands que 255 -habituellement des adresses mémoire- dans deux positions mémoire adjacentes et que nous stockons d'abord l'octet qui contient les unités.

Ainsi, pour stocker le nombre #4026 dans les emplacements mémoire #0402 et #0403 nous utilisons :

```
POKE #0402, #26
POKE #0403, #40
```

(dans la machine V1.0 nous avons à convertir #26 et #40 en 38 et 64 respectivement).

Pour retrouver le nombre et l'imprimer nous pourrions utiliser :

```
PRINT PEEK(#0402)+256*PEEK(#0403)
```

ou si nous traitons un résultat hexadécimal :

```
PRINT HEX$(PEEK(#0402)+256*PEEK(#0403))
```

ORIC fournit des méthodes plus faciles à l'aide des instructions DOKE (ce qui signifie Double pOKE) et DEEK (ce qui signifie Double pEEK).

Pour faire l'équivalent de ces opérations en utilisant les instructions DOKE et DEEK on aurait :

```
DOKE #0402,#4026
PRINT DEEK(#0402)
PRINT HEX$(DEEK(#0402))
```

INSTRUCTION DEFINIE PAR L'UTILISATEUR (!)

! peut être utilisé comme une instruction BASIC utilisant une routine écrite en code machine. L'adresse du début de cette routine est mise par l'instruction DOKE dans l'emplacement mémoire #2F5 (c'est-à-dire stocké dans #2F5 et #2F6). ! peut alors être utilisé dans un programme BASIC.

Par exemple, le programme suivant transfère 255 octets mémoire des emplacements #700 à #7F2 dans les emplacements #900 à #9FE.

```
        LDX #FF
LOOP    LDA 6FF,X
        STA 8FF,X
        DEX
        BNE LOOP
        RTS
```

L'assemblage de ce programme donne le code machine suivant :

```
#A2 #FF
#BD #FF #06
#9D #FF #08
#CA
#D0 #F7
#60
```

Pour définir ! le programme est :

```
10 FOR N=0 TO 11
20 READ A
30 POKE #400+N,A
40 NEXT
50 DOKE #2F5,#400
60 DATA #A2,#FF
70 DATA #BD,#FF,#06
80 DATA #9D,#EF,#08
90 DATA #CA
100 DATA #D0,#F7
110 DATA #60
```

! peut être utilisé comme partie d'un programme ou comme une commande directe. Comparez le temps pris par la routine en code machine appelée par ! avec le temps que cela prendrait pour transférer 255 emplacements mémoire en utilisant PEEK et POKE.

FONCTIONS DEFINIES PAR L'UTILISATEUR

&(X) où X est un entier compris entre 0 et #FFFF, permet d'utiliser une routine écrite en code machine. Pour définir &(X) nous stockons l'adresse de début de la routine dans les emplacements mémoire #2FC et #2FD en utilisant l'instruction DOKE.

&(X) peut être alors utilisé dans n'importe quelle commande. Par exemple :

```
PRINT &(6)
IF &(0)>10 THEN ...
```

USR permet aussi à l'utilisateur de définir des fonctions mais d'une manière bien plus souple et bien plus générale que &(X).

USR met une valeur dans une zone réservée en mémoire utilisée par l'ordinateur pour les opérations arithmétiques et connue sous le nom d'accumulateur en virgule flottante. Ceci permet à la fonction USR d'accepter des nombres réels et de fournir un résultat qui soit un nombre réel. Les opérations en nombres réels sont plus difficiles à écrire en code machine que les instructions en nombres entiers.

La routine définie par USR est écrite en langage machine et USR est définie grâce à l'instruction :

```
DEF USR add
```

où add est l'adresse de la première instruction en code machine.

L'instruction :

```
PRINT USR(X)
```

mettra la valeur X dans l'accumulateur en virgule flottante, exécutera les opérations définies par le code des instructions machine commençant à l'adresse add, extraira le résultat numérique de l'accumulateur en virgule flottante et l'imprimera à l'écran.

USR et & utilisent normalement des routines en langage machine qui se trouvent dans la mémoire ROM de l'ordinateur. Ces routines font partie du système d'exploitation de la machine.

Nous n'avons pas assez de place ici pour une étude détaillée du langage assembleur 6502 ou du système d'exploitation d'ORIC. Si vous souhaitez étudier le premier plus en profondeur, il y a de nombreux livres disponibles. Parmi les plus utiles il y a le livre de Rodnay Zaks (Sybex), et celui de Lance Leventhal (Osborne McGraw-Hill).

CHARGEMENT ET SAUVEGARDE D'INSTRUCTIONS LANGAGE MACHINE

Les programmes écrits en langage machine peuvent être sauvegardés et chargés à partir de bande grâce aux commandes CSAVE et CLOAD utilisant des blocs de mémoire. Par exemple :

```
CSAVE"MCP",A#400,E#420
CLOAD"MCP",A#400,E#420.
```

Les programmes en code machine peuvent être chargés dans les emplacements #400 à #420 sans affecter aucun programme BASIC déjà contenu dans la mémoire réservée à l'utilisateur. Si, toutefois, des programmes en langage machine doivent être chargés dans le haut de la mémoire réservée à l'utilisateur, l'instruction HIMEM doit être utilisée préalablement.

Chapitre 17

EN GUISE DE CONCLUSION

L'utilisation d'un ordinateur familial est facile. Certains livres font croire que cela est difficile. L'une des raisons principales en est que certains livres sont écrits par des professeurs afin de démontrer à d'autres professeurs combien ils sont intelligents..!

Je crois que la plupart des choses peuvent être expliquées simplement en utilisant un langage direct. Quelques termes techniques sont nécessaires mais le jargon doit être évité. Le jargon est une mauvaise graine qu'il est difficile de déraciner et je m'excuse par avance d'en avoir glissé quelque peu dans mon livre.

J'ai fait l'hypothèse que le lecteur n'avait pas besoin de connaissances spécialisées préalables. Si vous avez quelques connaissances en musique par exemple, vous avez trouvé que le chapitre 11 était simpliste. Est-ce que les connaissances de base en général sont si faibles que cela ? Oui, elles le sont pour la plupart des gens ! Beaucoup trop d'auteurs expliquent en long, en large et en travers des instructions BASIC très simples, mais font l'hypothèse que leurs lecteurs peuvent résoudre des équations mathématiques compliquées et connaissent par coeur des symphonies musicales.

Qu'allez-vous faire maintenant ? Si vous avez maîtrisé ce livre vous avez laissé loin derrière vous votre statut de débutant. Vous êtes maintenant un adepte et vous avez toute la connaissance nécessaire pour écrire des programmes utiles et élégants. Mais vous n'êtes pas encore un expert, cela nécessite de la pratique.

Pour votre développement et votre enrichissement personnel, je recommande que vous vous abonniez régulièrement à une ou deux revues de microinformatique. Tous les programmes qui apparaissent dans de telles revues ne sont pas particulièrement bons, mais d'une manière ou d'une autre, vous y trouverez des suggestions intéressantes ici et là. Mais ne suivez pas à la lettre les programmes que vous y trouverez, changez-les et améliorez-les, envoyez vos propres programmes aux magazines. Vous ne deviendrez pas riche grâce à cela, mais cette distraction présente bien des attraits !

Annexe A

CODES ASCII

Code ASCII (en décimal)	Code ASCII (en hexadécimal)	Caractère normal	Caractère alternatif
32	# 20	espace	
33	# 21	!	
34	# 22	"	
35	# 23	#	
36	# 24	$	
37	# 25	%	
38	# 26	&	
39	# 27	'	
40	# 28	(	
41	# 29	)	
42	# 2A	*	
43	# 2B	+	
44	# 2C	,	
45	# 2D	–	
46	# 2E	.	
47	# 2F	/	
48	# 30	0	
49	# 31	1	
50	# 32	2	
51	# 33	3	
52	# 34	4	
53	# 35	5	
54	# 36	6	
55	# 37	7	
56	# 38	8	
57	# 39	9	
58	# 3A	:	
59	# 3B	;	
60	# 3C	<	
61	# 3D	=	
62	# 3E	>	
63	# 3F	?	
64	# 40	@	

Code ASCII (en décimal)	Code ASCII (en hexadécimal)	Caractère normal	Caractère alternatif
65	# 41	A	
66	# 42	B	
67	# 43	C	
68	# 44	D	
69	# 45	E	
70	# 46	F	
71	# 47	G	
72	# 48	H	
73	# 49	I	
74	# 4A	J	
75	# 4B	K	
76	# 4C	L	
77	# 4D	M	
78	# 4E	N	
79	# 4F	O	
80	# 50	P	
81	# 51	Q	
82	# 52	R	
83	# 53	S	
84	# 54	T	
85	# 55	U	
86	# 56	V	
87	# 57	W	
88	# 58	X	
89	# 59	Y	
90	# 5A	Z	
91	# 5B	[	
92	# 5C	/	
93	# 5D	]	
94	# 5E	↑	
95	# 5F	£	
96	# 60	©	—
97	# 61	a	—
98	# 62	b	—
99	# 63	c	—
100	# 64	d	—
101	# 65	e	—
102	# 66	f	—
103	# 67	g	—
104	# 68	h	—
105	# 69	i	—
106	# 6A	j	—

Code ASCII (en décimal)	Code ASCII (en hexadécimal)	Caractère normal	Caractère alternatif
107	# 6B	k	—
108	# 6C	l	—
109	# 6D	m	—
110	# 6E	n	—
111	# 6F	o	—
112	# 70	p	—
113	# 71	q	—
114	# 72	r	—
115	# 73	s	—
116	# 74	t	—
117	# 75	u	—
118	# 76	v	—
119	# 77	w	—
120	# 78	x	—
121	# 79	y	—
122	# 7A	z	—
123	# 7B	{	—
124	# 7C	|	—
125	# 7D	}	—

Annexe B

BASIC ORIC

MOT CLEF	DESCRIPTION	FORMAT
ABS	Une fonction donnant la valeur absolue de son argument	ABS(N)
AND	L'opération du ET logique sur deux éléments et ce bit à bit. Egalement utilisé dans l'instruction IF...THEN.	X=#80 AND Y IF... AND... THEN...
ASC	Une fonction donnant la valeur décimale du code ASCII du premier caractère de la chaîne donnée comme argument de cette fonction. Si la chaîne est vide un code erreur est généré.	ASC(N$)
ATN	Une fonction donnant la valeur de l'arc tangente de l'angle exprimé en radiant, argument de la fonction	ATN(A)
CALL	Une instruction du programme qui en transfère le contrôle à une routine écrite en langage machine et qui commence à l'adresse spécifiée par l'argument.	CALL addr
CHAR	Une instruction qui place un caractère sur l'écran en mode HAUTE RESOLUTION. Le coin supérieur gauche se trouve à l'emplacement du curseur. X=code ASCII du caractère S=0 pour un caractère standard 1 pour un caractère alternatif. FB 0 à 3 voir ci-après.	CHAR X,S,FB

MOT CLEF	DESCRIPTION	FORMAT
CHR$	Une fonction qui crée une chaîne de caractère de longueur 1 contenant le caractère ASCII spécifiée par l'argument.	CHR$(42)
CIRCLE	Une instruction qui trace un cercle sur l'écran en mode HAUTE RESOLUTION. Le centre du cercle est déterminé par la position actuelle du curseur. R est le rayon (de 1 à 99) FB 0 à 3 -voir ci-dessous.	
CLEAR	Une instruction qui efface toutes les variables déclarées dynamiquement y compris les chaînes de caractères.	CLEAR
CLS	Une instruction qui efface la partie texte de l'écran, remet la couleur de l'arrière-plan à sa valeur courante et repositionne le pointeur à la première position normale pour le texte.	CLS
FB 0 arrière-plan 1 avant-plan 2 inversion 3 n rien faire		

MOT CLEF	DESCRIPTION	FORMAT
CLOAD	Une commande qui charge un nouveau programme depuis une cassette. Normalement l'ancien programme est effacé et toutes les variables sont effacées. CLOAD,S optionnel,vitesse 300 bauds CLOAD,V vérification (machine V1.1 seulement) CLOAD,J fusion (machine V1.1 seulement) CLOAD,A,E charge une partie spécifique de la mémoire.	CLOAD"PROG" CLOAD "PROG",S CLOAD "PROG",V CLOAD "PROG",S,V CLOAD "PROG",J CLOAD "PROG",S,J CLOAD "PROG",A# 500,E#600 CLOAD "PROG",A# 500,E#600,S
CONT	Une commande qui poursuit l'exécution d'un programme après un break.	CONT
COS	Une fonction qui donne le cosinus de son argument exprimé en radian.	COS(A)
CURMOV	Une instruction qui positionne le curseur sur une nouvelle position de l'écran en mode HAUTE RESOLUTION X et Y sont les déplacements relatifs par rapport à la position précédente. FB 0 à 3 (voir ci-dessus). La nouvelle position doit se trouver dans les limites de l'écran.	CURMOV X,Y,FB

MOT CLEF	DESCRIPTION	FORMAT
CURSET	Une instruction qui positionne le curseur aux coordonnées absolues X, Y sur l'écran en mode HAUTE RESOLUTION. X compris entre 0 et 239 Y compris entre 0 et 199. FB 0 à 3 (voir ci-dessus).	CURSET X,Y,FB
CSAVE	Une commande qui sauvegarde un programme dans un fichier sur cassette. CSAVE,S (optionnel, vitesse de sauvegarde 300 bauds).	CSAVE"PROG" CSAVE "PROG",S CSAVE "PROG",A# 400,E#420
DATA	Doit précéder toutes les données à lire par READ. Les chaînes de caractères requièrent des guillemets, si elles contiennent des espaces au début.	DATA 1,2, A,"BC"
DEEK	Une fonction qui fournit un nombre égal au contenu de l'octet dont l'adresse est indiquée en argument, plus 256 fois le contenu de l'octet suivant en mémoire.	DEEK(addr)
DEF FN	Une instruction qui définit une fonction numérique.	DEF FNF(X)= X*X+3
DEF USR	Une instruction qui définit l'adresse de début d'une routine utilisateur en langage machine qui définit une fonction.	DEF USR#410

MOT CLEF	DESCRIPTION	FORMAT
DIM	Une instruction qui dimensionne un tableau. Les tableaux sont dimensionnés par défaut à 10. Les tableaux qui doivent être chargés à partir de fichier sur cassettes (machine V1.1) doivent être dimensionnés avec DIM.	DIM B$(15,2)
DOKE	Une instruction qui stocke la valeur N dans 2 emplacements mémoires contigus A et A+1. INT(N/256) va dans l'emplacement A+1 et N-INT(N/256) va dans l'emplacement A.	DOKE A,N
DRAW	Une instruction qui trace une droite sur l'écran en mode HAUTE RESOLUTION de la position actuelle du curseur à celle indiquée par le déplacement relatif X,Y. FB vaut 0 à 3.	DRAW X,Y,FB
ELSE	Séparation dans une instruction. Si dans une instruction IF la condition est fausse (pas vérifiée) alors les instructions suivant le ELSE seront exécutées.	IF...THEN... ELSE
END	Une instruction qui arrête le programme et renvoie en mode direct.	END
EXP	Une fonction qui calcule e élevé à la puissance de l'argument.	EXP(N)

MOT CLEF	DESCRIPTION	FORMAT
EXPLODE	Une instruction qui produit un bruit prédéfini.	EXPLODE
FALSE	Une fonction qui a pour valeur 0.	FALSE
FILL	Une instruction qui remplit X rangées sur Y colonnes de caractères dont le code ASCII est N, pour l'écran HAUTE RESOLUTION. X compris entre 1 et 40 Y compris entre 1 et 200 N compris entre 0 et 127	FILL Y,X,N
FN	Mot réservé qui permet d'obtenir le résultat d'une fonction numérique prédéfinie.	FNX(3)
FOR	Une instruction qui marque le début d'une boucle FOR...NEXT.	FOR N=1 TO 6
GET	Une instruction qui balaye le clavier jusqu'à ce qu'une touche soit enfoncée.	GET A GET B$
GOSUB	Une instruction qui transfère le contrôle du programme à une ligne numéro L ; mais permet de revenir par RETURN à l'instruction suivant immédiatement le GOSUB.	GOSUB L
GOTO	Une instruction qui transfère le contrôle du programme à une ligne numéro L.	GOTO L

MOT CLEF	DESCRIPTION	FORMAT
GRAB	Une commande qui alloue les emplacements mémoire. #9800 à #B400 (machine 48k) ou #1800 à #3400 (machine 16k) au programme de l'utilisateur.	GRAB
HEX$	Une fonction qui donne une chaîne de caractères contenant la valeur en hexadécimal du nombre N.	HEX$(N)
HIMEM	Une pseudo-variable qui fixe l'adresse mémoire maximale utilisable par l'interpréteur.	HIMEM#97FF
HIRES	Une instruction qui met l'écran en mode HAUTE RESOLUTION, met l'arrière-plan en noir, l'avant plan en blanc et positionne le curseur en 0,0 (coin supérieur gauche).	HIRES
IF	Une instruction qui établit un test à partir duquel on contrôle les actions suivantes du programme.	IF...THEN...
INK	Une instruction qui contrôle la couleur d'avant plan de tout l'écran. N est un entier compris entre 0 et 7.	INK N
INPUT	Une instruction qui arrête l'exécution du programme jusqu'à ce que la touche ENTER soit enfoncée Elle lit alors le contenu du tampon du clavier dans les variables spécifiées. Un message à l'écran, optionnel, peut être généré.	INPUT A INPUT B$ INPUT"...";A INPUT"..."; B$

MOT CLEF	DESCRIPTION	FORMAT
INT	Une fonction donnant l'entier immédiat inférieur ou égal à l'argument.	INT(N)
KEY$	Une fonction qui balaye le clavier sans arrêter le programme et fournit comme résultat une chaîne contenant le caractère correspondant à la touche enfoncée. Si aucune touche n'est enfoncée une chaîne nulle est générée.	A$=KEY$
LEFT$	Une fonction qui fournit les N caractères de gauche de la chaîne donnée en argument.	LEFT$ (B$,N)
LEN	Une fonction qui fournit la longueur en caractères de la chaîne donnée comme argument.	LEN(B$)
LET	Instruction d'affectation (optionnelle).	LET K=6
LIST	Une commande qui liste à l'écran le programme actuel ou des lignes spécifiées.	LIST LIST 120 LIST 10-60
LLIST	Fonctionne comme LIST mais l'impression est faite sur imprimante.	LLIST LLIST 120 LLIST 10-60
LN	Une fonction qui donne le logarithme naturel de son argument.	LN(X)
LOG	Une fonction donnant le logarithme à base 10 de son argument.	LOG(X)

MOT CLEF	DESCRIPTION	FORMAT
LORES	Une instruction qui bascule l'écran en mode BASSE RESOLUTION. La couleur d'arrière plan est noire et la couleur d'avant plan blanche. Le curseur est positionné dans le coin supérieur droit.	LORES 0 LORES 1
LPRINT	Une instruction qui imprime un message spécifique à l'imprimante.	LPRINT 6 LPRINT "SALUT"
MID$	Une fonction qui fournit N caractères à partir de la position M de la chaîne de caractères.	MID$ (A$,M,N)
MUSIC	Une instruction qui spécifie une note musicale ou un son à jouer à l'aide de PLAY. CH est compris entre 1 et 6 et spécifie le canal OCT. OCT est compris entre 0 et 6 et spécifie l'octave. NT est compris entre 1 et 12 et spécifie la note. VOL est compris entre 1 et 15 et spécifie le volume. Quand VOL vaut 0, le volume est contrôlé par PLAY.	MUSIC CH, OCT,NT,VOL
NEW	Une commande qui initialise l'interprèteur en vue d'un nouveau programme.	NEW
NEXT	Une instruction marquant la fin d'une boucle FOR...NEXT.	NEXT NEXT N NEXT I,J
NOT	Une opération logique équivalente à la noégation booléenne.	NOT(X) NOT(A=B)

MOT CLEF	DESCRIPTION	FORMAT
ON	Une instruction qui débranche le programme à l'une des adresses indiquées selon la valeur d'une variable.	ON N GOSUB.. ON N GOTO...
OR	L'opération logique de l'union entre 2 éléments fonctionne bit à bit. Peut aussi être utilisé dans l'instruction IF...THEN.	X=#80 or Y IF...OR... THEN
PAPER	Une instruction qui contrôle la couleur d'arrière-plan de tout l'écran. N est un entier compris entre 0 et 7.	PAPER N
PATTERN	Une instruction qui initialise le registre PATTERN. N est un entier compris entre 0 et 255.	PATTERN N
PEEK	Une fonction qui donne le contenu de l'emplacement mémoire X.	PEEK(X)
PI	Une fonction qui vaut 3.14159265.	PI
PING	Une instruction qui produit un son prédéfini.	PING
PLAY	Une instruction qui produit un son défini par SOUND et MUSIC. TE est compris entre 0 et 7 et spécifie les canaux de la musique. NE est compris entre 0 et 7 et spécifie les canaux du bruit. EM est compris entre 0 et 7 et spécifie l'enveloppe. EP est compris entre 0 et 32767 et spécifie la période de l'enveloppe.	PLAY TE, NE, EM, EP

MOT CLEF	DESCRIPTION	FORMAT
PLOT	Une instruction qui place un caractère ou une chaîne de caractère sur l'écran en mode texte ou BASSE RESOLUTION. Un argument numérique est interprété comme un code ASCII.	PLOT 7,10, "A" PLOT 8,4,A$ PLOT 9,14,82
POINT	Une fonction qui vaut -1 si le pixel spécifié par X et Y est de la couleur d'arrière plan et qui vaut 0 s'il est de la couleur d'avant plan.	POINT (X,Y)
POKE	Une instruction qui met la valeur N (compris entre 0 et 255) dans l'emplacement mémoire A.	POKE A,N
POP	Une instruction qui enlève du haut de la pile l'adresse de retour généré par l'appel le plus récent à un sous programme.	POP
POS	Une fonction qui fournit la position horizontale du curseur dans l'écran en mode TEXTE ou BASSE RESOLUTION. Pour la machine V1.1 POS(0) fournit la position du curseur sur l'écran. POS(1) fournit la position de la tête de l'imprimante.	POS
PRINT	Une instruction qui affiche un message à l'écran.	PRINT1 PRINT"SALUT"
READ	Une instruction qui affecte à des variables des valeurs lues dans des instructions DATA du programme.	READ A READ B$

MOT CLEF	DESCRIPTION	FORMAT
RECALL	Une instruction qui charge dans un tableau, préalablement dimensionné, les variables stockées dans un fichier sur cassette par l'instruction STORE. Ne fonctionne que sur la machine V1.1.	RECALL A, "FICHIER" RECALL B$ "FICHIER" RECALL A, "FICHIER",S
RELEASE	Une instruction qui assigne à l'écran en mode HAUTE RESOLUTION la portion de mémoire affectée antérieurement au programme utilisateur par l'instruction GRAB.	RELEASE
REM	Une instruction qui fait que le reste de la ligne est ignoré.	REM
REPEAT	Une instruction qui marque le début d'une boucle REPEAT...UNTIL.	REPEAT
RESTORE	Une instruction qui repositionne le pointeur du READ sur la première donnée des instructions DATA.	RESTORE
RETURN	Une instruction qui renvoie à l'instruction qui suit le dernier GOSUB exécuté.	RETURN
RIGHT$	Une fonction qui donne les N caractères de droite de la chaîne fournie comme argument.	RIGHT$(A$,N)

MOT CLEF	DESCRIPTION	FORMAT
RND	Une fonction qui génère un nombre pseudo aléatoire compris entre 0 et 0.999999999. Si X>=1 un nombre aléatoire différent est généré à chaque fois. Si X = 0 le même nombre aléatoire est généré chaque fois. Si X < 0 le générateur de nombre au hasard est fixé de telle manière qu'il produise à chaque fois la même série de nombres pseudo aléatoires.	RND(X)
RUN	Une commande exécutant le programme à partir de la ligne spécifiée, ou à partir de la première ligne si rien n'est spécifié. Toutes les variables définies dynamiquement (y compris le tableaux) sont réinitialisées.	RUN RUN N
SCRN	Une fonction qui donne la valeur du code ASCII du caractère se trouvant aux coordonnées X,Y lorsque l'écran est en mode texte ou basse résolution.	CHAR(X,Y)
SGN	Une fonction qui vaut -1 si l'argument est négatif, 0 si l'argument est nul, +1 si l'argument est positif.	SGN(X)
SHOOT	Une instruction produisant un son prédéfini.	SHOOT
SIN	Une fonction donnant le SINUS de son argument en radian.	SIN(A)

MOT CLEF	DESCRIPTION	FORMAT
SOUND	Une instruction décrivant le son à jouer par l'instruction PLAY. C est compris entre 1 et 6 et spécifie le canal. P est compris entre 1 et 4096 et spécifie la période. V est compris entre 0 et 15 et spécifie le volume. Quand V vaut 0 le volume est contrôlé par l'instruction PLAY.	SOUND C,P,V
SPC	Une fonction qui utilisée avec PRINT ou LPRINT permet d'imprimer N espaces.	SPC(N)
SQR	Une fonction qui donne la racine carrée de son argument.	SQR(X)
STEP	Cette fonction spécifie des pas autres que 1 dans l'instruction FOR.	FOR...TO... STEP N
STOP	Une instruction qui arrête un programme, qui peut alors être relancée par CONT.	STOP
STORE	Une commande qui stocke les variables d'un tableau préexistant dans un fichier sur cassette (seulement pour machine V1.1), S optionnel, vitesse de l'opération en bauds : 300.	STORE A, "FICHIER" STORE B$ "FICHIER" STORE A "FICHIER",S
STR$	Une fonction qui donne le caractère ASCII correspondant au nombre donné comme argument.	STR$(N)

MOT CLEF	DESCRIPTION	FORMAT
TAB	Une fonction qui, utilisée avec PRINT ou LPRINT, déplace de début de l'impression de N caractères sur la gauche (ne fonctionne correctement que sur la machine V1.1).	TAB(N)
TAN	Une fonction qui donne la tangente de son argument exprimé en radian.	TAN(X)
TEXT	Une commande basculant l'écran en mode texte.	TEXT
THEN	Un mot clef utilisé avec IF afin de décider une action du programme.	IF...THEN...
TROFF	Une instruction désactivant la fonction de TRACE.	TROFF
TRON	Une instruction activant la fonction de TRACE.	TRON
TRUE	Une fonction donnant -1.	TRUE
USR	Une fonction fournissant son argument à une routine en virgule flottante et récupérant le résultat de cette routine.	DEF USR=addr USR(N)

MOT CLEF	DESCRIPTION	FORMAT
VAL	Une fonction donnant le code ASCII de son argument	VAL(B$)
WAIT	Une instruction qui arrête le programme pendant un certain nombre de centième de seconde.	WAIT N
ZAP	Une instruction produisant un son prédéfini.	ZAP

Annexe C

OCCUPATION MEMOIRE

OCCUPATION MEMOIRE POUR 48 K

	MODE TEXTE ET BASSE RESOLUTION		**MODE HAUTE RESOLUTION**	
FFFF				FFFF
	Mémoire morte (ROM)		Mémoire morte (ROM)	
C000				C000
	Libre		Libre	
BFE0				BFE0
	Ecran		Ecran	
BB80				
	Ensemble de Caracteres alternatifs			
B800				
	Ensemble des caractères standards			A000
B400			Ensemble des caractères alternatifs	
	Programme utilisateur (si la commande GRAB est donnée)			9C00
			Ensemble des caractères standards	
9800				9800
	Programme utilisateur		Programme utilisateur	
0500				0500
	Page 4 (jusqu'à 420 pour les programmes de contrôle mémoire)		Page 4 (jusqu'à 420 pour les programmes de contrôle mémoire)	
0400				0400
	Page 3 (adresses des E/S physiques)		Page 3 (adresses des E/S physiques)	
0300				0300
	Page 2 (variables dynamiques)		Page 2 (variables dynamiques)	
0200				0200
	Page 1 (pile)		Page 1 (pile)	
0100				0100
	Page 0 (allouée)		Page 0 (allouée)	
0000				0000

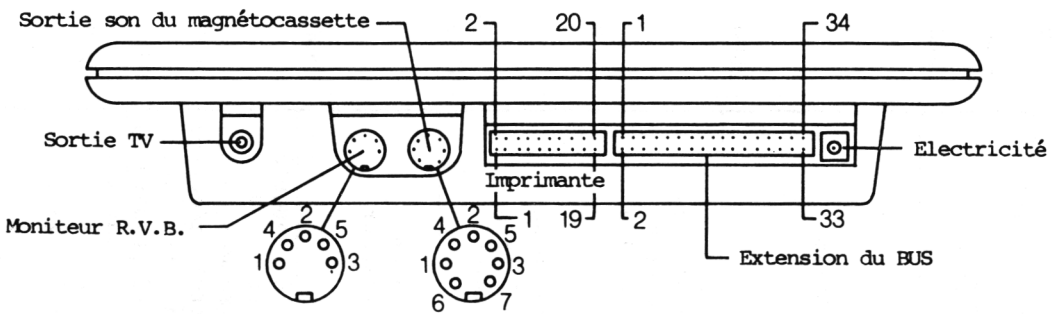
(Tous les nombre sont en hexadécimal)

OCCUPATION MEMOIRE POUR 16k

	MODE TEXTE ET BASSE RESOLUTION	MODE HAUTE RESOLUTION	
FFF			FFFF
	Mémoire morte (ROM)	Mémoire morte (ROM)	
C000			C000
	Non exploité	Non exploité	
4000			4000
	Libre	Libre	
3FE0			3FE0
	Ecran	Ecran	
3B80			
	Ensemble des caractères alternatifs		
3800			
	Ensemble des caractères standards		2000
3400		Ensemble des caractères alternatifs	
	Programme utilisateur (si commande GRAB)		1C00
		Ensemble des caractères standards	
1800			1800
	Programme utilisateur	Programme utilisateur	
0500			0500
	Page 4 (jusqu'à 420 pour les programmes de contrôle mémoire)	Page 4 (jusqu'à 420 pour les programmes de contrôle mémoire)	
0400			0400
	Page 3 (adresses des E/S physiques)	Page 3 (adresses des E/S physiques)	
0300			0300
	Page 2 (variables dynamiques)	Page 2 (variables dynamiques)	
0200			0200
	Page 1 (pile)	Page 1 (pile)	
0100			0100
	Page 0 (allouée)	Page 0 (allouée)	
0000			0000

Annexe D

DESCRIPTION DES BROCHES DES PRISES



EXTENSION DU BUS

MAP	1	2	ROMDIS
02	3	4	RESET
E/S	5	6	CONTROLE DES E/S
R/W	7	8	IRQ
D2	9	10	D0
A3	11	12	D1
A0	13	14	D6
A1	15	16	D3
A2	17	18	D4
D5	19	20	A4
A5	21	22	D7
A6	23	24	A15
A7	25	26	A14
A8	27	28	A13
A9	29	30	A12
A10	31	32	A11
+5V	33	34	TERRE

IMPRIMANTE

STB	1	2	TERRE
D0	3	4	TERRE
D1	5	6	TERRE
D2	7	8	TERRE
D3	9	10	TERRE
D4	11	12	TERRE
D5	13	14	TERRE
D6	15	16	TERRE
D7	17	18	TERRE
ACK	19	20	TERRE

R.V.B

1. ROUGE
2. VERT
3. BLEU
4. SYNCHRO
5. TERRE

CASSETTE

1. Enregistrement
2. TERRE
3. Lecture
4. Son
5. Son
6. Contact pour le contrôle du moteur
7. Contact pour le contrôle du moteur

Annexe E

MESSAGES D'ERREURS

Les messages d'erreurs sont affichés avec un numéro de ligne quand ils se produisent lors de l'exécution d'un programme.

1) **CAN'T CONTINUE**
Tentative de relancer un programme, après qu'une instruction ait été ajoutée ou enlevée après le STOP ou après un CTRL C.

2) **DISP TYPE MISMATCH**
Une instruction a été utilisée dans le mauvais contexte (par exemple CHAR en mode TEXTE).

3) **DIVISION BY ZERO**
La division par zéro donne l'infini et crée donc un débordement de capacité.

4) **FORMULA TOO COMPLEX**
Plus de deux instructions **IF/THEN** sont utilisées dans la même ligne.

5) **ILLEGAL DIRECT**
Une instruction, qui ne peut être utilisée que dans un programme, a été utilisée comme une commande donnée à partir du clavier (par exemple DATA).

6) **ILLEGAL QUANTITY**
ORIC devait calculer l'incalculable (par exemple racine carrée de -1).

7) **NEXT WITHOUT FOR**
Une instruction NEXT sans instruction FOR correspondante.

8) **OUT OF DATA**
Une tentative a été faite de lire (READ) plus de données qu'il n'y en a de définies par les instructions DATA du programme.

9) **OUT OF MEMORY**
Plus de mémoire disponible pour l'utilisateur.
Mais peut également être causé par plus de 16 boucles imbriquées ou 16 sous programmes s'appelant.

10) OVERFLOW
Dépassement de capacité. Le programme essaie de calculer un nombre plus grand que 1 70141*10^{38} .

11) REDIM'D ARRAY
Tentative de dimensionner à nouveau un tableau déjà dimensionné.

12) RETURN WITHOUT GOSUB
Une instruction RETURN est rencontrée sans que le programme ait préalablement exécuté l'instruction GOSUB correspondante.

13) STRING TOO LONG
Une chaîne de plus de 254 caractères a été spécifiée.

14) BAD SUBSTRING
Référence est faite à un élément de tableau qui n'existe pas : la valeur d'un des indices de l'élément est supérieur aux dimensions du tableau.

15) SYNTAX ERROR
Erreur de syntaxe : une instruction a été mal orthographiée ou incorrectement ponctuée.

16) TYPE MISMATCH
Une tentative a été faite pour assigner un nombre dans une chaîne de caractères ou inversement.

17) UNDEF'D STATEMENT
Une tentative a été faite pour sauter à un numéro de ligne qui n'existe pas.

18) UNDEF'D FUNCTION
Une tentative a été faite pour utiliser une fonction qui n'a pas été définie au préalable.

19) REDO FROM STARD
Un caractère non numérique a été entré au clavier lorsque l'instruction INPUT voulait initialiser une variable numérique.

20) BAD UNTIL
Le programme rencontre un UNTIL sans avoir préalablement exécuté le REPEAT correspondant.

Annexe F

ENSEMBLE DES INSTRUCTIONS EN ASSEMBLEUR 6502

ADC Ajouter un emplacement mémoire à l'accumulateur avec retenue.

AND ET logique d'un emplacement mémoire avec l'accumulateur.

ASL Décalage à gauche d'un bit (mémoire ou accumulateur).

BCC Branchement s'il n'y a pas de retenue.

BCS Branchement s'il y a une retenue.

BEQ Branchement si le résultat est nul.

BIT Teste les bits d'un emplacement mémoire avec ceux de l'accumulateur.

BMI Branchement si le résultat est négatif.

BNE Branchement si le résultat n'est pas nul.

BPL Branchement si le résultat est positif.

BRK Arrêt.

BVC Branchement en cas de remise à zéro du bit de dépassement de capacité.

BVS Branchement en cas de positionnement du bit de dépassement de capacité.

CLC Remise à zéro du bit de retenue.

CLD Remise à zéro du mode décimal.

CLI Remise à zéro du bit de bloquage des interrupteurs.

CLV Remise à zéro du bit de dépassement de capacité.

CMP Compare un emplacement mémoire et l'accumulateur.

CPX Compare un emplacement mémoire et l'indice X.

CPY Compare un emplacement mémoire et l'indice Y.

DEC Soustrait 1 d'un emplacement mémoire.

DEX Soustrait 1 de l'indice X.

DEY Soustrait 1 de l'indice Y.

EOR OU exclusif d'un emplacement mémoire avec l'accumulateur.

INC Ajoute 1 à un emplacement mémoire.

INX Ajoute 1 à l'indice X.

INY Ajoute 1 à l'indice Y.

JMP Saut à la nouvelle adresse.

JSR Saut à la nouvelle adresse en sauvegardant l'adresse de retour.

LDA Initialise l'accumulateur à partir d'un emplacement mémoire.

LDX Initialise l'indice X à partir d'un emplacement mémoire.

LDY Initialise l'indice Y à partir d'un emplacement mémoire.

LSR Décalage à droite d'un bit (mémoire ou accumulateur)

NOP Pas d'opération.

ORA OU entre un emplacement mémoire et l'accumulateur.

PHA Mettre le contenu de l'accumulateur sur la pile.

PHP Mettre le registre d'état sur la pile.

PLA Initialiser le contenu de l'accumulateur à partir de la pile.

PLP Initialiser le contenu du registre d'état à partir de la pile.

ROL Rotation d'un bit sur la gauche (emplacement mémoire ou accumulateur).

ROR Rotation d'un bit sur la droite (emplacement mémoire ou accumulateur).
RTI Retour d'interruption.
RTS Retour de sous-programme.
SBC Soustrait le contenu d'un emplacement mémoire de l'accumulateur avec retenue.
SEC Positionne le bit de retenue.
SED Positionne le mode décimal.
SEI Positionne le bit de blocage d'interruption.
STA Copie le contenu de l'accumulateur dans un emplacement mémoire.
STX Copie le contenu de l'index X dans un emplacement mémoire.
STY Copie le contenu de l'index Y dans un emplacement mémoire.
TAX Transfère le contenu de l'accumulateur dans l'index X.
TAY Transfère le contenu de l'accumulateur dans l'index Y.
TSX Transfère le pointeur de la pile dans l'index X.
TXA Transfère l'index X dans l'accumulateur.
TXS Transfère l'index dans le pointeur de pile.
TYA Transfère l'index Y dans l'accumulateur.

ADC Ajouter un emplacement mémoire à l'accumulateur avec retenue — NZCIDV
Opération:A+M+C-A,C — ///—/

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	ADC	#Oper	69	2	2
Page Zéro	ADC	Oper	65	2	3
Page Zéro,X	ADC	Oper,X	75	2	4
Absolu	ADC	Oper	6D	3	4
Absolu,X	ADC	Oper,X	7D	3	4*
Absolu,Y	ADC	Oper,Y	79	3	4*
(Indirect,X)	ADC	(Oper,X)	61	2	6
(Indirect),Y	ADC	(Oper),Y	71	2	5*

* Ajouter 1 si une limite de page est franchie

AND ET logique d'un emplacement mémoire avec l'accumulateur — NZCIDV
Opération:A ∧ M-A — //——

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	AND	#Oper	29	2	2
Page Zéro	AND	Oper	25	2	3
Page Zéro,X	AND	Oper,X	35	2	4
Absolu	AND	Oper	2D	3	4
Absolu,X	AND	Oper,X	3D	3	4*
Absolu,Y	AND	Oper,Y	39	3	4*
(Indirect,X)	AND	(Oper,X)	21	2	6
(Indirect),Y	AND	(Oper),Y	31	2	5*

* Ajouter 1 si une limite de page est franchie

ASL Décalage à gauche d'un bit NZCIDV
Opération:C-76543210-0 (mémoire ou accumulateur) ///---

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Accumulateur	ASL	A	0A	1	2
Page Zéro	ASL	Oper	06	2	5
Page Zéro,X	ASL	Oper,X	16	2	6
Absolu	ASL	Oper	0E	3	6
Absolu,X	ASL	Oper,X	1E	3	7

BCC Branchement s'il n'y a pas NZCIDV
Op.:Branchement si C=0 de retenue ------

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Relatif	BCC	Oper	90	2	2*

* Ajouter 1 si on débranche dans la même page
* Ajouter 2 si on débranche dans une autre page

BCS Branchement s'il y a NZCIDV
Op.:Branchement si C=1 une retenue ------

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Relatif	BCS	Oper	B0	2	2*

* Ajouter 1 si on débranche dans la même page
* Ajouter 2 si on débranche dans la page suivante

BEQ Branchement si le NZCIDV
Op.:Branchement si Z=1 Résultat est nul ------

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Relatif	BEQ	Oper	F0	2	2*

* Ajouter 1 si on débranche dans la même page
* Ajouter 2 si on débranche dans la page suivante

BIT Test des bits d'un emplacement N ZCIDV
Op.:a ∧ M,M_7-N,M_6-V mémoire avec l'accumulateur M_7/——M_6

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Page Zéro	BIT	Oper	24	2	3
Absolu	BIT	Oper	2C	3	4

BMI Branchement si le résultat NZCIDV
Op.:Branchement si N=1 est négatif ——

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Relatif	BMI	Oper	30	2	2*

* Ajouter 1 si on débranche dans la même page
* Ajouter 2 si on débranche dans une autre page

BNE Branchement si le résultat NZCIDV
Op.:Branchement si Z=0 n'est pas nul ——

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Relatif	BNE	Oper	D0	2	2*

* Ajouter 1 si on débranche dans la même page
* Ajouter 2 si on débranche dans une autre page

BPL Branchement si le résultat NZCIDV
Op.:Branchement si N=0 est positif ——

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Relatif	BPL	Oper	10	2	2*

* Ajouter 1 si on débranche dans la même page
* Ajouter 2 si on débranche dans une autre page

BRK Arret NZCIDV
Op.:Interruption forcée PC+2P ---*--

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	BRK	00	1	7

* Une commande BRK ne peut etre masquée par un positionnement I.

BVC Branchement si remise à zéro du bit de dépassement de capacité NZCIDV
Op.:Branchement si V=0 ------

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Relatif	BVC Oper	50	2	2*

* Ajouter 1 si on débranche dans la même page
* Ajouter 2 si on débranche dans une autre page

BVS Branchement si positionnement du bit de dépassement de capacité NZCIDV
Op.:Branchement si V=1 ------

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Relatif	BVS Oper	70	2	2*

* Ajouter 1 si on débranche dans la même page
* Ajouter 2 si on débranche dans une autre page

CLC Remise à zéro du bit de retenue NZCIDV
Opération:0-C --0---

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	CLC	18	1	2*

CLD — Remise à zéro du mode décimal — NZCIDV
Opération:0-D — ----0-

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	CLD	D8	1	2

CLI — Remise à zéro du bit de blocage des interruptions. — NZCIDV
Opération:0-I — ---0--

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	CLI	58	1	2

CLV — Remise à zéro du bit de dépassement de capacité — NZCIDV
Opération:0-V — -----0

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	CLV	B8	1	2

CMP — Compare un emplacement mémoire et l'accumulateur — NZCIDV
Opération:A-M — ///---

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Immédiat	CMP #Oper	C9	2	2
Page Zéro	CMP Oper	C5	2	3
Page Zéro,X	CMP Oper,X	D5	2	4
Absolu	CMP Oper	CD	3	4
Absolu,X	CMP Oper,X	DD	3	4*
Absolu,Y	CMP Oper,Y	D9	3	4*
(Indirect,X)	CMP (Oper,X)	C1	2	6
(Indirect),Y	CMP (Oper),Y	D1	2	5*

* Ajouter 1 si une limite de page est franchie

CPX Compare un emplacement mémoire et NZCIDV
Opération:X-M l'indice X ///--

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	CPX	#Oper	E0	2	2
Page Zéro	CPX	Oper	E4	2	3
Absolu	CPX	Oper	EC	3	4

CPY Compare un emplacement mémoire et NZCIDV
Opération:Y-M l'indice Y ///--

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	CPY	#Oper	C0	2	2
Page Zéro	CPY	Oper	C4	2	3
Absolu	CPY	Oper	CC	3	4

DEC Soustrait 1 d'un emplacement NZCIDV
Opération:M1-M mémoire //---

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Page Zéro	DEC	Oper	C6	2	5
Page Zéro,X	DEC	Oper,X	D6	2	6
Absolu	DEC	Oper	CE	3	6
Absolu,X	DEC	Oper,X	DE	3	7

DEX Soustrait 1 de l'indice X NZCIDV
Opération:X-1X //---

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	DEX	CA	1	2

DEY Soustrait 1 de l'indice Y NZCIDV
Opération:Y1-Y //----

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	DEY	88	1	2

EOR OU exclusif d'un emplacement mémoire avec l'accumulateur NZCIDV
Opération:AM-A //----

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	EOR	#Oper	49	2	2
Page Zéro	EOR	Oper	45	2	3
Page Zéro,X	EOR	Oper,X	55	2	4
Absolu	EOR	Oper	4D	3	4
Absolu,X	EOR	Oper,X	5D	3	4*
Absolu,Y	EOR	Oper,Y	59	3	4*
(Indirect,X)	EOR	(Oper,X)	41	2	6
(Indirect),Y	EOR	(Oper),Y	51	2	5*

* Ajouter 1 si une limite de page est franchie

INC Ajoute 1 à un emplacement mémoire NZCIDV
Opération:M+1-M //----

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Page Zéro	INC	Oper	E5	2	5
Page Zéro,X	INC	Oper,X	F5	2	6
Absolu	INC	Oper	ED	3	6
Absolu,X	INC	Oper,X	FD	3	7

INX Ajoute 1 à l'indice X NZCIDV
Opération:X+1-X //----

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	INX	E8	1	2

INY Ajoute 1 à l'indice Y NZCIDV
Opération:Y+1-Y //----

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Implicite	INY		C8	1	2

JMP Saut à la nouvelle adresse NZCIDV
Opération: (PC+1)-PCL ------
(PC+2)-PCH

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Absolu	JMP	Oper	4D	3	3
Indirect	JMP	(Oper)	6C	3	5

JSR Saut à la nouvelle adresse NZCIDV
Opération:(PC+2),(PC+1)-PCL avec sauvegarde de l'adresse ------
(PC+2)-PCH de retour

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Absolu	JSR	Oper	20	3	6

LDA Inititialise l'accumulateur à NZCIDV
Opération:M-A partir d'un emplacement mémoire //----

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	LDA	#Oper	A9	2	2
Page Zéro	LDA	Oper	A5	2	3
Page Zéro,X	LDA	Oper,X	B5	2	4
Absolu	LDA	Oper	AD	3	4
Absolu,X	LDA	Oper,X	BD	3	4*
Absolu,Y	LDA	Oper,Y	B9	3	4*
(Indirect,X)	LDA	(Oper,X)	A1	2	6
(Indirect),Y	LDA	(Oper),Y	B1	2	5*

* Ajouter 1 si une limite de page est franchie

LDX Initialise l'indice X à partir NZCIDV
Opération:M-X d'un emplacement mémoire //----

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	LDX	#Oper	A2	2	2
Page Zéro	LDX	Oper	A6	2	3
Page Zéro,X	LDX	Oper,X	B6	2	4
Absolu	LDX	Oper	AE	3	4
Absolu,X	LDX	Oper,X	BE	3	4*

* Ajouter 1 si une limite de page est franchie

LDY Initialise l'indice Y à partir NZCIDV
Opération:M-Y d'un emplacement mémoire //----

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	LDY	#Oper	A0	2	2
Page Zéro	LDY	Oper	A4	2	3
Page Zéro,X	LDY	Oper,X	B4	2	4
Absolu	LDY	Oper	AC	3	4
Absolu,X	LDY	Oper,X	BC	3	4*

* Ajouter 1 si une limite de page est franchie

LSR Décalage à droite d'un bit NZCIDV
Opération:0-76543210-C (mémoire ou accumulateur) 0//---

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Accumulateur	LSR	A	4A	1	2
Page Zéro	LSR	Oper	46	2	5
Page Zéro,X	LSR	Oper,X	56	2	6
Absolu	LSR	Oper	4E	3	6
Absolu,X	LSR	Oper,X	5E	3	7

NOP Pas d'opération NZCIDV
Opération:Pas d'opération (2 cycles) ------

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	NOP	EA	1	2

ORA OU entre un emplacement mémoire et l'accumulateur NZCIDV
Opération:AVM-A //----

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	ORA	#Oper	09	2	2
Page Zéro	ORA	Oper	05	2	3
Page Zéro,X	ORA	Oper,X	15	2	4
Absolu	ORA	Oper	0D	3	4
Absolu,X	ORA	Oper,X	1D	3	4*
Absolu,Y	ORA	Oper,Y	19	3	4*
(Indirect,X)	ORA	(Oper,X)	01	2	6
(Indirect),Y	ORA	(Oper),Y	11	2	5*

* Ajouter 1 si une limite de page est franchie

PHA Mettre le contenu de l'accumulateur sur la pile NZCIDV
Opération:A ------

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	PHA	48	1	3

PHP Mettre le registre d'état sur la pile NZCIDV
Opération:P ------

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	PHP	08	1	3

PLA Initialiser l'accumulateur à partir de la pile NZCIDV
Opération:A //——

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	PLA	68	1	4

PLP Initialiser le registre d'état à partir de la pile NZCIDV
Opération:P Pile

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	PLP	28	1	4

ROL Rotation d'un bit sur la gauche (mémoire ou accumulateur) NZCIDV
Opération:76543210-C* ///——

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Accumulateur	ROL	A	2A	1	2
Page Zéro	ROL	Oper	26	2	5
Page Zéro,X	ROL	Oper,X	36	2	6
Absolu	ROL	Oper	2E	3	6
Absolu,X	ROL	Oper,X	3E	3	7

* M ou A

ROR Rotation d'un bit sur la droite (mémoire ou accumulateur) NZCIDV
Opération:C-76543210* ///——

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Accumulateur	ROR	A	6A	1	2
Page Zéro	ROR	Oper	66	2	5
Page Zéro,X	ROR	Oper,X	76	2	6
Absolu	ROR	Oper	6E	3	6
Absolu,X	ROR	Oper,X	7E	3	7

* M ou A

RTI Retour d'interruption NZCIDV
Opération:P PC Pile

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	RTI	40	1	6

RTS Retour de sous-programme NZCIDV
Opération:PC,PC+1-PC ------

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	RTS	60	1	6

SBC Soustrait le contenu d'un emplacement mémoire de avec retenue NZCIDV
Opération:A-M-C-A ///--/
NB: C=Retenue

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Immédiat	SBC	#Oper	E9	2	2
Page Zéro	SBC	Oper	E5	2	3
Page Zéro,X	SBC	Oper,X	F5	2	4
Absolu	SBC	Oper	ED	3	4
Absolu,X	SBC	Oper,X	FD	3	4*
Absolu,Y	SBC	Oper,Y	F9	3	4*
(Indirect,X)	SBC	(Oper,X)	E1	2	6
(Indirect),Y	SBC	(Oper),Y	F1	2	5*

* Ajouter 1 si une limite de page est franchie

SEC Positionne le bit de retenue NZCIDV
Opération:1-C --1---

Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	SEC	38	1	2

SED Positionne le mode décimal NZCIDV
Opération:1-D ----1-

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Implicite	SED		F8	1	2

SEI Positionne le bit de blocage d'interruption NZCIDV
Opération:1-I ---1--

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Implicite	SEI		78	1	2

STA Copie le contenu de l'accumulateur dans un emplacement mémoire NZCIDV
Opération:A-M ------

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Page Zéro	STA	Oper	85	2	3
Page Zéro,X	STA	Oper,X	95	2	4
Absolu	STA	Oper	8D	3	4
Absolu,X	STA	Oper,X	9D	3	5
Absolu,Y	STA	Oper,Y	99	3	5
(Indirect,X)	STA	(Oper,X)	81	2	6
(Indirect),Y	STA	(Oper),Y	91	2	6

STX Copie le contenu de l'indice X dans un emplacement mémoire NZCIDV
Opération:X-M ------

Mode d'adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Page Zéro	STX	Oper	86	2	3
Page Zéro,Y	STX	Oper,Y	96	2	4
Absolu	STX	Oper	8E	3	4

STY Copie le contenu de l'indice Y NZCIDV
Opération:Y-M dans un emplacement mémoire ------

Mode d' adressage	Format en assembleur		Code Op.	Nombre Octets	Nombre Cycles
Page Zéro	STY	Oper	84	2	3
Page Zéro,X	STY	Oper,X	94	2	4
Absolu	STY	Oper	8C	3	4

TAX Transfère le contenu de NZCIDV
Opération:A-X l'accumulateur dans l'indice X //----

Mode d' adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	TAX	AA	1	2

TAY Transfère le contenu de NZCIDV
Opération:A-Y l'accumulateur dans l'indice Y //----

Mode d' adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	TAY	A8	1	2

TSX Transfère le pointeur de la pile NZCIDV
Opération:S-X //----

Mode d' adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	TSX	BA	1	2

TXA Opérateur:X-A	Transfère l'indice X dans l'accumulateur			NZCIDV //----
Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	TXA	8A	1	2

TXS Opérateur:X-S	Transfère l'index X dans le pointeur de pile			NZCIDV ------
Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	TXS	9A	1	2

TYS Opérateur:Y-A	Transfère l'index Y dans l'accumulateur			NZCIDV ------
Mode d'adressage	Format en assembleur	Code Op.	Nombre Octets	Nombre Cycles
Implicite	TYA	98	1	2

00-BRK
01-ORA (Indirecte,X)
02-Usage futur
03-Usage futur
04-Usage futur
05-ORA-Page zéro
06-ASL-Page zéro
07-Usage futur
08-PHP
09-ORA-Immédiat
0A-ASL-Accumulateur
0B-Usage futur
0C-Usage futur
0D-ORA-Absolu
0E-ASL-Absolu
0F-Usage futur
10-BPL
11-ORA (Indirecte),Y
12-Usage futur
13-Usage futur
14-Usage futur
15-ORA-Page zéro,X
16-ASL-Page zéro,X
17-Usage futur
18-CLC
19-ORA-Absolu,Y
1A-Usage futur
1B-Usage futur
1C-Usage futur
1D-ORA-Absolu,X
1E-ASL-Absolu,X
1F-Usage futur
20-JSR
21-AND (Indirecte,X)
22-Usage futur
23-Usage futur
24-BIT-Page zéro
25-AND-Page zéro
26-ROL-Page zéro
27-Usage futur
28-PLP
29-AND-Immédiat
2A-ROL-Accumulateur
2B-Usage futur
2C-BIT-Absolu
2D-AND-Absolu
2E-ROL-Absolu
2F-Usage futur
30-BMI
31-AND (indirecte),Y
32-Usage futur
33-Usage futur
34-Usage futur
35-AND-Page zéro,X
36-ROL-Page zéro,X
37-Usage futur
38-SEC
39-AND-Absolu,Y
3A-Usage futur
3B-Usage futur
3C-Usage futur
3D-AND-Absolu,X
3E-ROL-Absolu,X
3F-Usage futur
40-RTI
41-EOR (Indirecte,X)
42-Usage futur
43-Usage futur
44-Usage futur
45-EOR-Page zéro
46-LSR-Page zéro
47-Usage futur
48-PHA
49-EOR-Immédiat
4A-LSR-Accumulateur
4B-Usage futur
4C-JMP-Absolu
4D-EOR-Absolu
4E-LSR-Absolu
4F-Usage futur
50-BVC
51-EOR (Indirecte),Y
52-Usage futur
53-Usage futur
54-Usage futur
55-EOR-Page zéro,X
56-LSR-Page zéro,X
57-Usage futur
58-CLI
59-EOR-Absolu,Y
5A-Usage futur
5B-Usage futur
5C-Usage futur
5D-EOR-Absolu,X
5E-LSR-Absolu,X
5F-Usage futur
60-RTS
61-ADC (Indirecte,X)
62-Usage futur
63-Usage futur
64-Usage futur
65-ADC-Page zéro
66-ROR-Page zéro
67-Usage futur
68-PLA
69-ADC-Immédiat
6A-ROR-Accumulateur
6B-Usage futur
6C-JMP-Indirecte
6D-ADC-Absolu
6E-ROR-Absolu
6F-Usage futur
70-BVS
71-ADC (Indirecte),Y
72-Usage futur
73-Usage futur
74-Usage futur
75-ADC-Page zéro,X
76-ROR-Page zéro,X
77-Usage futur
78-SEI
79-ADC-Absolu,Y
7A-Usage futur
7B-Usage futur
7C-Usage futur
7D-ADC-Absolu,X
7E-ROR-Absolu,X
7F-Usage futur
89-Usage futur
81-STA (Indirecte,X)
82-Usage futur
83-Usage futur
84-STY-Page zéro
85-STA-Page zéro
86-STX-Page zéro
87-Usage futur
88-DEY
89-Usage futur
8A-TXA
8B-Usage futur
8C-STY-Absolu

8D-STA-Absolu
8E-STX-Absolu
8F-Usage futur
90-BCC
91-STA (Indirecte),Y
92-Usage futur
93-Usage futur
94-STY-Page zéro,X
95-STA-Page zéro,X
96-STX-Page zéro,Y
97-Usage futur
98-TYA
99-STA-Absolu,Y
9A-TXS
9B-Usage futur
9C-Usage futur
9D-STA-Absolu,X
9E-Usage futur
9F-Usage futur
A0-LDY-Immédiat
A1-LDA (Indirecte,X)
A2-LDX-Immédiat
A3-Usage futur
A4-LDY-Page zéro
A5-LDA-Page zéro
A6-LDX-Page zéro
A7-Usage futur
A8-TAY
A9-LDA-Immédiat
AA-TAX
AB-Usage futur
AC-LDY-Absolu
AD-LDA-Absolu
AE-LDX-Absolu
AF-Usage futur
B0-BCS
B1-LDA (Indirecte),Y
B2-Usage futur.
B3-Usage futur
B4-LDT-Page zéro,X
B5-LDA-Page zéro,X
B6-LDX-Page zéro,Y
B7-Usage futur
B8-CLV
B9-LDA-Absolu,Y
BA-TSX
BB-Usage futur
BC-LDY-Absolu,X
BD-LDA-Absolu,X
BE-LDX-Absolu,Y
BF-Usage futur
C0-CPY-Immédiat
C1-CMP-(Indirecte,X)
C2-Usage futur
C3-Usage futur
C4-CPY-Page zéro
C5-CMP-Page zéro
C6-DEC-Page zéro
C7-Usage futur
C8-INY
C9-CMP-Immédiat
CA-DEX
CB-Usage futur
CC-CPY-Absolu
CD-CMP-Absolu
CE-DEC-Absolu
CF-Usage futur
D0-BNE
D1-CMP-(Indirecte),Y
D2-Usage futur
D3-Usage futur
D4-Usage futur
D5-CMP-Page zéro,X
D6-DEC-Page zéro,X
D7-Usage futur
D8-CLD
D9-CMP-Absolu,Y
DA-Usage futur
DB-Usage futur
DC-Usage futur
DD-CMP-Absolu,X
DE-DEC-Absolu,X
DF-Usage futur
E0-CPX-Immédiat
E1-SBC-(Indirecte,X)
E2-Usage futur
E3-Usage futur
E4-CPX-Page zéro
E5-SBC-Page zéro
E6-INC-Page zéro
E7-Usage futur
E8-INX
E9-SBC-Immédiat
EA-NOP
EB-Usage futur
EC-CPX-Absolu
ED-SBC-Absolu
EE-INC-Absolu
EF-Usage futur
F0-BEQ
F1-SBC-(Indirecte),Y
F2-Usage futur
F3-Usage futur
F4-Usage futur
F5-SBC-Page zéro,X
F6-INC-Page zéro,X
F7-Usage futur
F8-SED
F9-SBC-Absolu,Y
FA-Usage futur
FB-Usage futur
FC-Usage futur
FD-SBC-Absolu,X
FE-INC-Absolu,X
FF-Usage futur

INDEX

ABS 129
accidentels 156
accumulateur en virgule flottante 239
accumulateur 232
activation du bruit 144
alerte de remplissage du clavier 26
AND 104,140
animation 174-175
antenne aérienne 3
arithmétique 87-88
arrêt 50
ASC 125
assembleur 231
ATN 132
attributs haute résolution 191-192
attributs 31-36,91-92,97-99 191-192
AUTO 60

baud 56
boucle continue 111
boucle infinie 68
boucles imbriquées 109
bouton de sélection du canal 5

canal du son 143
caractères alternatifs 33,164-166
caractères de contrôle 27-30,90-91
caractères double hauteur 5 34-35
caractères par ligne 217
cassettes 53
chaînes de caractères 76-77,117-126
CHAR 194-195
chargement du code machine 240
CHR$ 89,126
CIRCLE 185
clavier 7
CLEAR 127
clef de fa 150
clef de sol 150
cliquetis du clavier 12
CLOAD 54-55
CLS 70
code machine 230-240
code objet 234
code opération 233
codes de contrôle de l'imprimante 219-220
coloris globaux en mode haute résolution 184
commandes en mode immédiat 37
commandes implicites 220-227
comparaison de chaînes de caractères 120
compilateur 230
compteur d'enregistrement 52
compteur ordinal 232-233
concaténation 118
CONT 67
contrôle du moteur de cassette 52
COS 132,197
courbes 197-200
CSAVE 54
CURMOV 179-180
CURSET 179
curseur graphique 178
curseur 6

DATA 133-134

déballage de l'ordinateur 2
décision utilisant des chaînes de caractères 119
DEEK 237
DEF FN 132-133
défilement à l'écran 69
DEL (la touche) 26
demitonalité 153
dépôt du code machine en mémoire 235-236
dessus de la pile 208
deux points (:) 66
DIM 134-136
DIN (connecteurs) 52-53
DOKE 237
DRAW 180-181
durée des notes 156-157

écran mode haute résolution 178-179
EDIT 41
END 115
enregistreur de cassettes 52
entrée de programmes 37-39
enveloppe du son 146
erreurs dans les tableaux et chaînes de caractères 210
ESCAPE (le caractère) 91-93
EXP 131
EXPLODE 142

FALSE 207
FB (les nombres) 179-180
fichier 202
FILL 180-190
FN 132-133
fonctions définies par l'utilisateur 168-173,196
FOR...NEXT 71,107-109
franges d'interférences 183
fusion de programmes 58-59
gamme 152
GET 124-125
GOTO 67
GRAB 205
graphiques définis par l'utilisateur 168-173,196

harmonie 151
hauteur du son 152
HEX$ 140
HIMEM 210,236-237
HIRES 177-179

IF...THEN 103
IF...THEN...ELSE 104-107
ILLEGAL QUANTITY ERROR 125
impression de caractères composites 219
impression de graphiques 220-227
impression de texte 218-220
imprimante de l'ORIC 217-227
imprimantes (autres) 228-229
INK 72
INPUT 99-101
instructions définies par l'utilisateur 238
INT 129
interpréteur 131
interrupteur RESET 13
interruption AFT 9
interruptions 210

jargon 241

KEY$ 124

langage assembleur 233-235
LEFT$ 122
LEN 122
LET 71
lignes d'adresses 210
lignes de données 210
ligne contenant plusieurs instructions 106
LIST 39,70
LLIST 218
LN 131
LOG 131,199
LORES 0 97
LORES 1 166-167
LPRINT 218-227

manipulation des données 201-202
manipulation des fichiers 202-205
mémoire contenant les caractères 169,173,196,207
mémoire de l'écran 206
mémoire de l'écran en mode haute résolution 192-193
microprocesseur 231-233

MID$ 123
mode d'impression 218
modification d'une instruction 40-47
MUSIC 148-149

nettoyer l'écran 11
nombres hexadécimaux 139-140
nombres linéaires 137-139
NOT 104
numéro de ligne 39
numérotation des lignes 47

octave 152
ON...GOSUB 115
ON...GOTO 115
OR 104,140

PAPER 72
PATTERN 186
PEEK 173
périphériques d'entrées/sorties 210-211
PI 131-132
pile 208
PING 142
pixel 168
PLAY 143-144
PLOT 95-96
point virgule (;) séparateur du PRINT 78-80
pointeur de la pile 232-233
POKE 173-174
POP 208
portée 152
portes 210
POS 208-209,229
PRINT 74-87
PRINT @ 86-87
prise d'alimentation électrique 3
prise d'antenne aérienne 3
problèmes avec l'enregistreur de cassettes 55
puissance 131
PULL 209

READ 133-134
RECALL 203
recherche de programmes 60
recursion 114
registre d'état 232-233
registre modèle 186-187
registres 231-233
registres d'index 232
RELEASE 205
REM 65-66
REPEAT...UNTIL 110-111
répétition automatique 26
RESTORE 133-134
RETURN (touche) 26
RIGHT$ 122
RND 129-130
rotation de la position de la plume 220
RTS 235
RUN 39-67

sauvegarde d'affichages à l'écran 205-206
sauvegarde de blocs mémoire 61,205-207
sauvegarde de longs programmes 61
sauvegarde d'ensemble de caractères 207
sauvegarde du code machine 240
SCRN 175-176
SGN 128-129
SHOOT 142
SIN 132,197
SOUND 144-146
source 234
sous-programmes 112-115
SPC 85
SQR 131
STEP 197
STOP 66
STORE 202
STR$ 121
suppression de lignes 47
surimpression 219
SYNTAX ERROR 9
système d'exploitation 230

tableaux 134-136,201-202
texte clignotant 33
tableaux d'entiers 136
tampon du clavier 26
théorie musicale 150-158
tableaux de réels 136
touches shiftées 25
tableau de chaînes de caractères 136

TAB 82-84
TAN 132
TEXT 97
TE (activation du son) 144
TROFF 116
TRON 116
TRUE 207
TYPE MISMATCH ERROR 125

USR 239

virgule (,) séparateur du PRINT 80-82
variables entières 126
vitesse d'impression 217
vérification des programmes 57
variables de chaînes de caractères 117,119,126
VAL 121
variables 80,126-127

WAIT 72

ZAP 142

Dépôt légal Septembre 1984
Imprimé en Grande-Bretagne

INTERMICRO Collection dirigée par Jacky Akoka

ORIC
Oric/Oric Atmos

L'auteur, Ian Mclean, membre du Microelectronics Educational Development Centre, Paisley College of Technology, Ecosse, a près de 20 ans d'expérience dans l'enseignement et l'industrie électronique.

Cet ouvrage présente aux débutants en informatique les concepts fondamentaux relatifs à l'utilisation du micro-ordinateur ***ORIC.***
Il permet notamment d'apprendre à manipuler les différentes composantes de l'ordinateur (clavier, écran, mémoires, imprimante) et ce, pour les deux versions les plus répandues : ***ORIC*** et ***ORIC-ATMOS.***
Le langage BASIC est exposé et illustré à l'aide de nombreux exemples.
Une attention particulière est consacrée aux fonctions sonores et graphiques de l'***ORIC.***

Ce livre ne requiert aucune connaissance préalable en informatique ou en mathématiques. Ecrit de façon simple et alerte, il vous introduit de manière progressive aux différentes possibilités offertes par l'***ORIC.***

LES POINTS FORTS

- Panorama complet des possibilités de l'***ORIC*** tant en ce qui concerne la première version (Oric) que la nouvelle (Oric-Atmos).
- Introduction graduelle au langage BASIC.
- Illustration des commandes de gestion de fichiers, chargement et sauvegarde inclus.
- Création de graphismes à l'écran en modes basse et haute résolution avec utilisation de l'imprimante graphique.
- Présentation des aspects relatifs à la gestion du son et à la composition sonore.
- Des dizaines d'exemples de programmes BASIC faciles à comprendre avec la possibilité de les modifier pour les adapter à ses propres besoins.

Prentice/Hall International

Inter Editions 87 avenue du Maine 75014 Paris

ISBN 2 7296 0112 0

F 115 / 84.7

Image système Graph 8
Couverture Studio TRACT

Ian Mc Lean

Votre ORIC

Oric/Oric Atmos